transnationale Universiteit Limburg
School voor Informatietechnologie

A User and Designer Perspective on Multimodal Interaction in 3D Environments

Proefschrift voorgelegd tot het behalen van de graad van
Doctor in de Wetenschappen, richting Informatica
aan de transnationale Universiteit Limburg
te verdedigen door

Joan De Boeck

Promotor: Prof. dr. Karin Coninx

2007

transnationale Universiteit Limburg
School voor Informatietechnologie

A User and Designer Perspective on Multimodal Interaction in 3D Environments

Proefschrift voorgelegd tot het behalen van de graad van
Doctor in de Wetenschappen, richting Informatica
aan de transnationale Universiteit Limburg
te verdedigen door

Joan De Boeck

Promotor: Prof. dr. Karin Coninx

2007

Acknowledgments

The research described in this thesis would never have been realised by oneself. Therefore, I want to thank all the people who have, in one or other aspect, contributed to the realisation of this thesis.

In a first place, my work has been greatly inspired by the people in the 'Human Computer Interfaces' research group I've been working in for several years. Therefore, I want to thank my promotor Prof. dr. Karin Coninx and my near colleagues Erwin Cuppens, Lode Vanacken, Davy Vanacken, Maarten Cardinaels, Tim Tutenel and Johan Huysmans. Kris Luyten, Pieter Jorissen, Chris Vandervelpen, Tim Clerckx, Jan Van den Bergh, Kristof Verpoorten and Luc Adriaens deserve a word of thanks, as well, for their valuable help during the past years. In particular I want to formulate a special word of thanks to Chris Raymaekers, Tom De Weyer and Fabian Di Fiore for their personal friendship, besides our professional relations.

I also want to thank the EDM and its management Prof. Dr. Eddy Flerackers and Prof. dr. Frank Van Reeth, for giving chances to young researchers to develop their talents and produce good work. Not forgetting Ingrid Konings and Roger Claes at the EDM secretariat, who were also of a great help to me for all the practical things they arranged during the past years.

Of course I have to thank all the (ex)colleagues at my other employment, respectively at CIT, PortaCapena and the Katholieke Hogeschool Kempen, to show understanding for the sometimes demanding periods during my research.

Finally, my life would have been completely different without the invaluable support and interest of my family: my father, Remi and my two sisters, Tine and Ellen, my uncle Willy Baute, but especially my mother, Anny Baute, who died last year, far too young.

Last, but certainly not least, I'm truly grateful to my companion Marijke Van Roey and my two lovely little sons Ruben and Niels to who this thesis is dedicated.

Abstract

Creating user interfaces that fully support the three dimensions, is not a simple process. Although 3D interfaces have been successfully applied in several domains, they often support a poor interaction, compared to the possibilities of our human body. Current solutions often limit to visual output (e.g. via a screen) and 2D or 3D input via mice or trackers. As a consequence, those 3D environments are often far from the intuitive and user friendly application they are intended to be.

In our daily life, humans have several possibilities to communicate, such as speech, the hearing, the vision, touch and gestures. *Seen from a user's perspective*, the research described in this thesis is focused on improving the interaction within such a 3D environment. Therefore, we describe several user experiments describing the possible benefits of force feedback, speech input and two-handed interaction.

Since finding the best interface for a 3D environment is often a process of trial-and-error, the development cycle of such applications is more often than not long and expensive. *Seen from a designer's perspective*, this thesis shows how the design process may be facilitated. We propose NiMMiT, a visual notation, that can be used to describe interaction with the environment as a part of a model-based development approach. In this context, we also show how an application framework can abstract from the physical devices, supporting easy testing of the environment.

We believe that the results of this thesis will contribute to an easier and more intuitive interaction in 3D environments.

Contents

List of Figures

Part I

Theoretical Foundations

Chapter 1

Introduction

1.1 Problem Statement

Over the last decades, virtual environments have proven their benefits in a number of applications. Realistic 3D representations are used in the design phase of automobiles or in military training applications, but also in architectural design or entertainment. The aim of 3D environments may be to explore a complex model, visualise 3D data or perform complex operations on the data, etc. In any case, the application supporting the 3D world must be highly interactive, and hence an intuitive user interface is of an utmost importance.

Traditional solutions are often limited to visual and auditory output, and the environment is mostly controlled by a standard PC mouse and keyboard, or better with a 3D mouse or a tracked glove. As it is known that people have five senses (sight, hearing, touch, smell and taste), all intensively used in our daily interaction with the real world, this kind of interaction with the 3D world is very limited. It may be clear that, in order to support complex tasks in these environments, applications should make optimal use of our human capabilities to communicate. Not only visual and auditory output, but also haptic feedback (the sense of 'feeling' the world) should be present. On the other hand, not only a 3D mouse or two gloves, but also speech input or proprioception (the knowledge of ones position of the limbs) can enrich the interaction.

In this thesis, we take as a starting point a personal desktop setup in which the user is sitting in front of a monitor or a projection screen. As in such a

desktop setup, other physical attributes, such as drawings, a sheet of paper may be used or a telephone call or a conversation with a colleague can happen, traditional immersive solutions like head mounted displays are not suitable. During previous research in our lab, we created a personal 'semi-immersive' setup forming the basis for the experiments in this text. The PSD (Personal Surround Display) [De Boeck et al., 2003a] provides a wider viewing angle using three synchronised non-stereo projection screens, each 90 cm wide, located with an angle of 60 degrees, as shown in figure 1.1. The projection transformation in the rendering pipeline of both side screens are adjusted in such a manner that all three screens have the same projection centre. A PHANToM Haptic Device is used for the interaction within the world, providing additional force feedback.

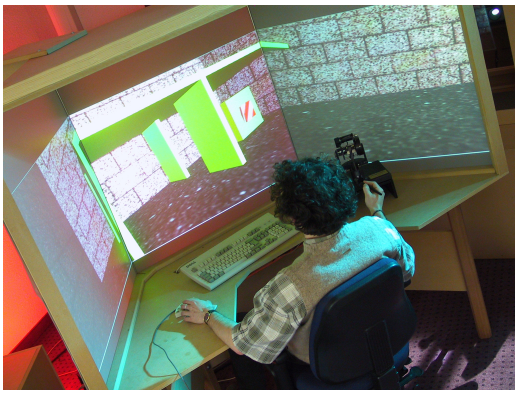


Figure 1.1: The Personal Surround Display

Although the environment provides a workable personal environment, some problems, which are also common in other traditional virtual environments, still exist. Those problems include difficulties to access objects which are placed at a certain 'depth' in the world, the limited workspace of the PHANToM device and limited access to widgets such as menus and dialogs. In spite of the improved visual output and additional force feedback, in some cases, some of those problems are even made stronger because of the large scaling factor between the size of the PSD and the limited workspace of the PHANToM. In this dissertation, we aim for a solution which addresses those problems by making optimal use of the user's 'communication' capabilities, obviously within the bounds of technical and budgetary limits. As the 'best' solution never exists, but solutions only may be 'better' or 'worse' dependent on the specific situation or context in which they are used, a designer of a 3D

environment typically spends a lot of time in the cycle of designing, implementing, testing and *re*designing a solution. This thesis will also consider at the development of a multimodal application from a designer's perspective. In this context, we will propose some solutions to facilitate the design process.

1.2 Thesis Overview

We start this part of the thesis by defining the most important terminology which will be used throughout this thesis. Here we also shortly describe some important theories on multimodal interaction. After that, we describe our vision on interaction in a virtual environment and consequently elaborate on user tasks, devices and metaphors. In the last chapter of this part, we propose NiMMiT, a notation which has been developed in the scope of this research, and which is optimised to describe multimodal interaction. This notation allows us to easily test and evaluate newly proposed interaction techniques.

Part II describes the research towards a multimodal interface. Each step on this way is evaluated by a formal or informal experiment. Chapter 5 starts by elaborating on haptic feedback in general and describes some experiments on haptic viewpoint manipulation. Next we investigate if speech input can be beneficial in a personal virtual environment. As we can conclude from literature that two-handed interaction is another natural way of interacting in the real world, chapter 7 elaborates on this topic, and finally shows how proprioception together with two-handed input and force feedback can solve some problems.

Part III of this thesis describes some of the technical aspects which where important to support our research. In chapter 8 we evaluate how a second haptic device increases the computational load, and hence limits the complexity of objects and the 3D scene. Next, this chapter also shows a more formal approach which can be applied to evaluate the correctness and performance of new haptic algorithms. Chapter 9 elaborates on the software design of a research framework which supports the experiments described in part II. As the design and implementation phase of a multimodal interface is more often than not a long and expensive process, chapter 10 proposes some solutions which may facilitate this process.

Chapter 11 summarises our conclusions and points up the contributions of this research. Finally, in chapter 12, reflecting on the results, we consider some aspects of future research.

Chapter 2

Multimodality

Contents

2.1 What is Multimodality

In literature on *Multimodal Systems*, it is notable that nearly all authors handle their own usage of the basic terms in this domain. For this reason, we start this chapter by defining those basic terms as they are used throughout this thesis. We will define what is covered by *Multimodal Interaction*, and what is not.

As this thesis deals with multimodal interaction, we first have to define what is meant by *multimodality*. Dependent on the discipline, slightly different definitions are used. In the context of the human sciences, a modality is often

defined as a sensory modality:

Definition 2.1 ***a (sensory) modality*** *is a perception via one of the three perception-channels*[1] *(visual, auditive, tactile) [Charwat, 1994].*

This should not be confused with the representational modalities defined by Arens and Hovy:

Definition 2.2 ***a (representational) modality*** *is a single mechanism by which to express information. E.g. Spoken and written natural language, tables, forms, maps, ... [Arens and Hovy, 1990]*

In this work, we take the assumption that we consider a dialog between two agents: one of the agents involved is a human, while the other is a machine. Although the main goal of this research is to contribute to the creation of intuitive human-computer interfaces, many of the theoretical statements will keep a natural human-human interaction in mind.

Definition 2.3 *We define* ***a dialog*** *as a bidirectional communication between two agents, by which information is exchanged. In this work we consider a human and a computing system.*

Definition 2.4 *In the general sense,* ***a multimodal system*** *supports communication through different* sensory modalities *while carrying* meaning *through a* representational modality*.*

Or, as Nigay says:

> "A multimodal system strives for meaning"
> [Nigay and Coutaz, 1993].

Less straightforward, is the definition of the terms *channel* and *medium*, as they differ strongly among the different authors. Hovy et al. define a medium as 'the hardware facility utilized by a modality to realize the expression of the information' and a channel as an 'Independent dimension of variation of a particular information carrier in a particular substrate' [Hovy and Arens, 1990]. Without going in too much detail on the latter definition, it is said

[1] Humans actually have five senses but in most current research on multimodality, taste and olfactory are not considered.

Table 2.1: Different human senses

Sensory Perception	Sense Organ	Medium or Sensory Modality
Sense of Sight	Eyes	Visual
Sense of Hearing	Ears	Auditive
Sense of Touch	Skin	Haptic
Sense of Smell	Nose	Olfactory
Sense of Taste	Tongue	Taste
Sense of Balance	Organ of Equilibrium	Haptic
Sense of Temperature	Skin	Haptic

that e.g. an icon's channels are its shape, color, position, orientation, etc, in relation to the background map on which it is placed. Charwat [Charwat, 1994] defines a perception channel as one of the 5 senses, and a communication channel as a connection between one sending source and one receiving sink, which are in the human body communication represented by the sense organs and the muscles. Bernsen [Bernsen, 1994] distinguishes three media: graphics, sound and touch, all different in their perceptual qualities (visual, auditory, tactile).

For the purpose of this document, we adopt Bernsen's definition of a medium, as well as Hovy's definition of a channel, although slightly reformulated by Bernsen:

Definition 2.5 *A **medium** is the physical carrier used to contain the information. As a human we can distinguish the five media corresponding to our senses: Visual, Auditive, Haptic, Olfactory and Taste.*

Note that the definition of a medium (according to Bernsen) and a sensory modality (according to Charwat) are closely related. Table 2.1 gives an overview of the human senses.

Definition 2.6 *A **channel** is a perceptual aspect of a medium which can be used to carry information in context. E.g. colour is a channel of the visual medium.*

Although it is a bit contra-intuitive making an analogy with human-human-interaction, we define input and output as follows:

Definition 2.7 ***Output** is all meaning that is* sent by the computer,

Definition 2.8 ***Input*** *is everything that is* sent by the human *to the computer.*

2.2 Theoretical Approaches

In this section, we give an overview of some important theoretical foundations that can be found in literature and which can serve as a basic understanding of multimodal interaction between a human and a computer.

2.2.1 A preliminary Taxonomy

A preliminary taxonomy is described in the work of Hovy and Arens [Hovy and Arens, 1990],[Arens and Hovy, 1990]. The authors try to find rules to define what modality is best suitable to present what. The basic idea is that specific rules, which are empirically seen as 'rather obvious', are generalized.

An example can be:

Ship's locations are presented on maps

is generalised as

Data duples (of which locations (of e.g. ships) are an example) are presented on planar modalities (such as graphs, tables and maps)

The taxonomy starts by defining a lexicon of terms which are necessary. Some definitions which are of importance in the scope of this thesis, already have been handled in section 2.1. Based upon these definition, the authors define tables with the most common generic modalities and the values for their properties: e.g. properties of a single beep, a picture or a spoken sentence.

In the next step, according to the presentation of the data, six characteristics are defined. For each of these characteristics, several rules are defined that allow to make a choice of the medium or channel.

In an example, the authors show how these rules can be used in order to find the best available modality for presenting 'Paris' as a flight destination. The available data are: the city's coordinates, the name Paris and a picture of the Eiffel Tower. If we follow the author's reasoning, according to the rules, coordinates can be presented using maps, pictures, tables and graphs. It can be motivated that tables are ruled out. This leaves graphs and maps

as possible modalities, but looking at the internal semantics of the modalities, only maps remain.

Although this work contains an interesting point of view, the value for this thesis is rather limited; except for the fact that it provides us with some clear definitions, which will be used throughout this text.

2.2.2 A taxonomy of output modalities

Bernsen defines 'multimodal information' as a combination of unimodal data. Based upon this starting point, he tries to achieve a taxonomy of unimodal expressions. Combining items in this taxonomy creates the multimodal output representation of information as they appear in multimodal user interfaces.

In this taxonomy, each unimodal piece of information is given five orthogonal properties: a modality can either be linguistic or non-linguistic, analogue or non-analogue, arbitrary or non-arbitrary, static or dynamic and it is sent over a medium.

- Linguistic representations are based upon the systems of meaning we know from natural human language. A written text is an example of a linguistic representation.

- Analogue representations are called those pieces of information, from which the representation shares an analogy with reality. For instance, a picture is analogue, but the roman script is not.

- Arbitrary are those representations that are arbitrarily chosen. This is in contrast to the representations that cannot be chosen arbitrarily because they are widely known. Beeps can be arbitrarily, while spoken language is clearly not.

- A static representation can be decoded by the user in any order desired and as long as desired. A picture for instance is static, while a spoken phrase is an example of a dynamic representation.

- The media which are considered in this output taxonomy are visual, auditive and haptic.

Summarized, we can state that a modality can be expressed as follows:

$$\begin{gathered}\forall M \in Modalities; \\ \exists! l \in \{Linguistic, Non-Linguistic\}; \exists! a \in \{Analogue, Non-Analogue\}; \\ \exists! r \in \{Arbitrary, Non-Arbitrary\}; \exists! d \in \{Dynamic, Static\}; \\ \exists! m \in \{Visual, Audtitive, Haptic\} : M = (l, a, r, d, m)\end{gathered} \tag{2.1}$$

or abbreviated:

$$\begin{gathered}\forall M \in Modalities; \\ \exists! l \in \{+Li, -Li\}; \exists! a \in \{+An, -An\}; \\ \exists! r \in \{+Ar, -Ar\}; \exists! d \in \{+Dyn, -Dyn\}; \\ \exists! m \in \{Vi, Au, Ha\} : M = (l, a, r, d, m)\end{gathered} \tag{2.2}$$

If we combine all possible combinations of the five orthogonal properties, and we filter out all combinations that do not make sense in practice, the following groups can be defined:

- Linguistic Modalities, such as hieroglyphs (which are analogue) and written text (which is non-analogue).
  Linguistic modalities are expressed as (+Li,*,*,*,*)

- Analogue Modalities, which contain all non-linguistic modalities with an analogy to reality such as pictures, audio recordings, ...
  (-Li, +An, *, *, *)

- Arbitrary Modalities, which are opposite to analogue modalities. Examples are arbitrary acoustics or arbitrary pulses.
  (-Li, -An, +Ar, *, *)

- Explicit Modalities; these are the modalities that are not linguistic, not arbitrary and not analogue. Most well known non-analogue pictograms (such as arrows) fall in this category.
  (-Li, -An, -Ar, *, *)

The above mentioned categories contain both static and dynamic modalities, transmitted over either the visual, auditory or the haptic channel.

This taxonomy is exclusively developed for describing output modalities, however, with some adaptation it can be transformed for suiting input modalities [Bernsen, 1995]. Indeed, the first three coordinates, expressing the nature of the information, remain unchanged. The media by which humans can express themselves are somewhat different than the media by which computers can send information. Essentially, Bernsen proposes to remove the Dynamic/non-Dynamic coordinate and to change the haptic medium by the kinesthetic. When taking the principles of enactive interfaces into account (section 2.2.5), we are not convinced by this adaptation. Although most human expressions are dynamic, there is some static possibility of 'giving meaning' as well, such as the person's body posture, facial expressions and even some arbitrary postures (such as a flat hand in front of the body, meaning 'stop!'). Furthermore it is unclear what is meant by the human 'visual' medium or what is the difference with the 'kinesthetic' medium. In this context, we can take a look at the human's 'semiotic function' containing the kinesthetic apparatus, the voice, and facial expressions (for as far as this is not considered by the kinesthetic apparatus).

In the scope of this thesis we will define the most important modalities used in this dissemination as follows:

- Speech input/output: (+Li, -An, -Ar, +Dyn, Au)
- Visual output: (-Li, *, -Ar, *, Vi)
- Aural output: (-Li, *, *, +Dyn, Au)
- Direct manipulation[2]: (-Li, -An, +Ar, +Dyn, Kin)
- Postures/Gestures: (-Li, +An, -Ar, *, Kin)

2.2.3 Characteristics of Multimodal Interaction

As we now have a basic knowledge of a 'single modality', multimodal interaction can now be classified according to the (temporal) relations between the different modalities. Sturm et al [Sturm et al., 2002] define two categories: **sequential multimodality** is when two different actions of the user (or the

[2]Direct manipulation is an interaction style which allows a user to directly manipulate (computer representations of) objects using rapid, reversible, incremental actions and feedback (source: Wikipedia 2006).

system) use a different modality, while they are subsequent with no overlap in time. For instance if an object is selected by direct manipulation and then deleted by a speech command. We speak of **simultaneous multimodality** when two actions use a different modality and when they overlap in time. If both actions provide a part of a single piece of information it is called **coordinated simultaneous multimodality**, e.g. pointing at an object and simultaneously speaking a voice command that operates on that object. Otherwise we speak of **non-coordinated simultaneous multimodality**. An example of the latter category may be 'answering a phone call while driving a car'.

A similar classification can be found in [Coutaz et al., 1995]. In addition, Coutaz et al. identify four properties which characterize four types of relationships between modalities: the CARE properties (**C**omplementarity, **A**ssignment, **R**edundancy, **E**quivalence). These properties are applicable both on input and on output, and apply to both the user and the system.

System-CARE Properties

The *System*-CARE properties [Coutaz et al., 1995], define how the system can receive or send information.

Complementarity: Modalities are complementary when all the modalities are necessary for completing the task, but each is carrying just a part of the information. A typical example is a spoken command that must be accompanied by a pointing gesture, to indicate the subject of the command.

Assignment: A modality is assigned if there is no other modality to execute the task.

Redundancy: Modalities are redundant if they have the same expressive power for the task (see equivalence) but all of them must be used.

Equivalence: Modalities are equivalent for completing a task if it is necessary and sufficient to choose *one* of them.

All relations can be **permanent** or **transient** and are **total** or **partial**. A relation is permanent if it is true in any state of the application, otherwise it

is transient. A relation is total if it applies to every task of the application, otherwise it is partial.

The system-CARE properties can be used to analyze and understand existing multimodal systems, or systems under design: they allow to check them for possible usability issues, as the choice of one of the CARE properties has an influence for the user. Indeed: **equivalence** provides a way to improve flexibility, as there can be chosen between the possible modalities. Equivalence has also its benefits to improve the robustness: if one modality is not reliable (because of a device breakdown, or speech input in a noisy environment) the other can take over. **Redundancy**, on the other hand, can degrade robustness since both modalities *must* be present. **Assignment** can be seen as a restrictive feature because it forces the user to apply that modality. Finally, **complementarity** may cause cognitive overload and synchronization problems. Obviously **transiency** and **partiality**, have the risk of making a system inconsistent to the user.

User-CARE Properties

Similar to the System-CARE properties, Coutaz et al. also define the *User*-CARE properties. The user properties describe the user's preference between the available modalities.

User Complementarity: If the user's choice is to use one modality for one aspect of the task, and another modality for another aspect.

User Assignment: If only one modality is used by the user, or when the user has a strong preference for that modality.

User Redundancy: If the user chooses to use more than one modality to express the same piece of information.

User Equivalence: If the user is indifferent in using one or another modality out of a preferred subset of available modalities.

It is clear that the system-CARE properties must match the user-CARE properties. The exercise of matching both ends during system design, allows a designer to make predictions about the usability of the system, later.

2.2.4 A Brief Semiotic View

One of the key points in our definition of a multimodal system (def. 2.4) is that the system carries *meaning*. This is important, because we are not clicking the mouse because of the sake of 'just clicking' it, but because the click has a certain meaning in the current context of the interface.

Semiotics is the science of sign systems within society. Signs are meant to carry meaning, by standing for something else than themselves. Indeed, pictures do not stand for the picture itself, but the picture carries a similarity with reality that is depicted, just like a sort-icon on the computer screen communicates to the user the possibility of 'executing a sort algorithm'. Or like Andersen says: *a sign stands for something to someone in some respect* [Andersen, 1992]; referring to the sender, the receiver and the context of the sign. Although semiotics are not explicitly used throughout the work presented in this thesis, it is valuable to briefly introduce some ideas in this section, as a (slightly philosophical) background.

One of the basic assumptions of semiotics is that we cannot 'not communicate' [Scalisi, 2001], saying that everything in the world communicates, either conscious or not: the clouds can 'tell' us if it's going to rain or not, and humans can tell each other stories. Andersen claims that the human is a *compulsive interpreter* [Andersen, 2000]. Humans cannot help making sense of everything they perceive. Applied to the domain of 'Human Computer Interaction' (HCI), this could mean that users not only interpret the interface they are supposed to use, but also that they make guesses about what goes 'behind'. Helping users by building a relevant and correct internal model would make them more self-supporting, when the interface breaks down.

Humans are also *compulsive talkers*, saying that verbal communication is what makes our society hang together. Humans describe their own and other's tasks and communicate it to others. In the same way, user interfaces should not only be interpretable, but they should also be verbalisable: users must be able to 'describe' what they do with the interface, in order to use it. In addition, there must be no conflict between the existing professional language and the interface. For example, if for some people the basic dimension for classifying e.g. 'tasks' is based upon their location, then classifying them according to a temporal sequence makes this system less verbalisable and hence less usable for those people [Andersen, 1990].

De Souza [De Souza, 1993] states that semiotic HCI defines interfaces as 'messages sent by the designers to the user'. This point of view can explain several

common interface errors, as signs are always interpreted by the receivers inside the context of their current semiotic system. If two semiotic systems are different, the interpretation of the same sign will differ too. And it is certainly true that the semiotic context of a designer is essentially different of this of a user. On the other hand, it also has been shown that humans apply social rules in their relations with computers and think about the machine as a conscious entity, refuting De Souza's initial statement. This means that the contradiction between the 'real sender' and the 'perceived sender' also can explain several interaction problems.

Codes also make an essential part of a communication: from pictographs to the words in a sentence. These codes have to be learned. HCI often tries to develop systems that require a minimum of learning, but still, the user inherently has to learn new codes. 'Ease of use' thus can be partially seen as 'ease of learn' and 'ease to remember'. Creating systems with analogies to things we already know is in many cases a good solution to achieve this goal. As a result, metaphors are often defined in order to transfer this previous knowledge to the new domain (chapter 3).

2.2.5 From Action To Enaction

From the previous section, we stated that humans are compulsive talkers, which implies that our knowledge must be verbalised in order to understand and use it. It may be clear however that there exists another important category of knowledge which is far less verbalisable. Examples of this knowledge are sculpting, dancing, or driving a car; the knowledge is based on the experience of the perceptual responses received when acting in the world. We call this 'Enactive Knowledge'

Definition 2.9 ***Enactive knowledge*** *is knowledge stored in the form of motor responses, acquired by 'doing'.*

This is in contrast to the symbolic or iconic knowledge, described before, as we know from pictures, language and mathematical symbols. Cadoz [Cadoz, 2005] defines iconic and symbolic as follows:

Definition 2.10 *Something is an* ***iconic representation*** *of something else when there are some directly observable analogies between them. Examples are drawings, pictures, etc.*

Definition 2.11 *Something is an **symbolic representation** of something else when the relation between the representation and the represented is arbitrary. E.g. symbols for writing, arrows, ...*

In our every-day interaction with objects and other humans we are able to perform three main functions: we can observe and understand the world, we can give meaning to our feelings and thoughts, and we can act. Luciani et al [Luciani et al., 2005] define these three basic functions:

Definition 2.12 *The **epistemic function** is the function through which the environment is known. This can be done via the visual, the aural or the haptic channel, for instance by looking at an object or listening to a sound in order to identify it.*

Definition 2.13 *The **semiotic function** refers to a symbolic action. This is everything we can do to give meaning: a spoken phrase, facial expressions and gestures. A typical example of the latter is pointing at something with the finger. People will look at the pointed object, and not to the finger.*

Definition 2.14 *The **ergotic**[3] **function** refers to everything in which there is a physical energetic interaction between the human and the object (or other human). For instance throwing away an object, tapping onto the table, or playing an instrument.*

The sight and the hearing (but also the smell and the taste), are dedicated to the epistemic function, as they are the senses we use to understand the environment. Our voice is purely semiotic, as it is a natural way to express ourselves. Our gestural channel, let's say, our body movements is the only channel which supports all functions:

- Epistemic, in the sense that we can feel an object for its roughness, its temperature,...
- Semiotic, as we can make gestures which refer to a symbolic meaning.
- Ergotic, as we use our body movements for physical labor.

In a man-object relation, Cadoz [Cadoz, 2005] now defines a first and a second order loop. The 'first order loop', the 'gesture-touch-loop', is the most direct

[3] Ergotic comes from the Greek word 'Ergon', which means Physical Work, Energy

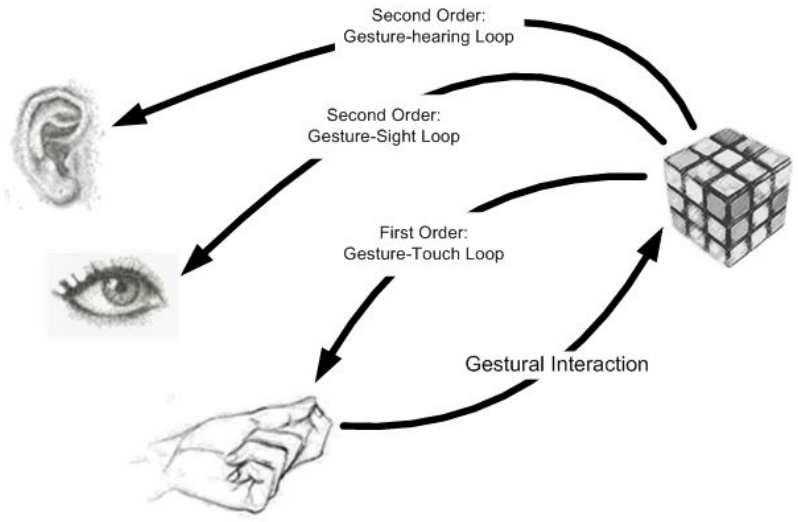


Figure 2.1: First and Second order loops when interacting with objects (according to [Cadoz, 2005])

one. The human is seen as a source of energy producing forces on a human-object system. The perceived feedback is of a very direct nature, and carries an energetic consistency with the generated force. When studying enactive knowledge, this loop is very important, as it is the first source of learning.

In the 'second order loop', the gesture-hearing or the gesture-sight loop, there is an energy transformation from the mechanical energy to a visual or acoustical phenomenon. The loops are depticted in figure 2.1.

One of the properties that distinguishes humans from most other animals, is the fact that we use instruments or tools in our daily work. The question rises wether we consider the human to interact with the tool, or the human to interact with the environment through the tool. Obviously, when we start learning e.g. to eat with fork and knife, or to play a musical instrument, we are concentrated on how to operate the tool or the instrument. When becoming an expert, we no more think about operating the tools or the instrument we use: we no more think about using fork and knife, and the musician can express the music as coming 'right from his body'. We speak of 'embodiement' when

we go from the first to the latter situation [Cadoz, 2005].

Definition 2.15 ***Embodiement*** *is the fact that an instrument or a tool becomes essentially part of the body, after a process of (enactive) learning. The operator will then use the tool or instrument as a part of its body.*

Things get more complex when humans start to operate machines. Until this point in the reasoning, there is an energetic consistency between the ergotic, semiotic and epistemic function, as the human is considered as the only source of energy. Machines are initially designed to be a source of external energy, breaking this energetic consistency. This results in an arbitrary link between the primary ergotic circuit, let's say between the human and the machine, and the secondary ergotic circuit between the machine and the environment. This is e.g. true when operating an excavator in which the sensory-motor loop is no longer applicable.

The last decades however, there is another shift which makes that machines (such as computers) are designed for providing information rather than for providing external energy. This makes it even more complex since the environment the user is interacting with, even has no material linkage anymore (e.g. a database system, or a VR application). Here lies the challenge for the researcher and the designer to 'reassess' the user's (everyday) enactive knowledge an allow him to apply it when interacting with this immaterial environment.

2.3 Summary

When discussing multimodality, several different definitions can be found in literature. Therefore, we started this chapter by defining some important terms, which will be used throughout this thesis. Thereafter, we have shortly explained some theories on multimodal interaction. Some of them, such as Bernsen's taxonomy, the CARE properties and the enactive point of view, in some respect will serve as a basis in this thesis. Hovy's work and the semiotic point of view on the other hand, are rather intended as a general background.

In the next chapter, we will focus on multimodality, specifically in 3D environments, where metaphors are used to transfer knowledge we already have from our daily life or from another domain to the new 3D interface.

Chapter 3

Multimodal Interaction in 3D Environments

Contents

3.1 Introduction

Performing tasks in a 3D or a virtual environment can be seen as a dialog between the user and the environment. As we know from definition 2.3, a dialog indicates a bidirectional exchange of information; in this case between the user who communicates his/her intentions to the computer, and the computer providing adequate feedback. Obviously, this task in 3D can become very complicated and hence the interface between the user and the computer must be 'easy to use' and 'easy to learn'. Currently, several '*interaction techniques*'(IT) have been described in literature, in order to make the execution of

the task as easy as possible. Very often the interaction technique is a *metaphor* (section 3.4) carrying an analogy with situations we already know from our daily life, or from other domains. In this way, a transfer is established between the known domain and the newly learned action. Since humans communicate multimodally by nature, interaction with the environment is also preferred to be multimodal. Therefore we can also state that the interaction technique may require one or more modalities. Finally, for the chosen modality, one or more input or output devices may be necessary.

We can summarise this reasoning as follows:

Figure 3.1: From task to physical device

In this figure, the arrow can be read as 'Makes use of a certain...'. The following example clarifies this:

- *A **Task** to be performed in the environment:* e.g. select an Object in the 3D world
- *The **Dialog** between the user and the computer:* e.g. the User has to point out the right coordinates and triggers the selection. The system gives feedback on the current coordinates and the current selection state.
- *The **Interaction Technique** or **Metaphor** to be used:* e.g. selection using 'Virtual Hand' (section 3.4.3)
- *The **Modalities** and the medium to be used:* e.g. direct Manipulation, using the kinesthetic apparatus for input; visual and haptic feedback as output.
- Possible **devices** to be used: e.g. magnetic Tracker, PHANToM Haptic device, Head Mounted Display, ...

In the remaining sections of this chapter we elaborate on the diagram shown in figure 3.1. We respectively go in detail on the tasks in 3D, the possible devices to support multimodality, and the different metaphors currently available.

3.2 Tasks in Virtual Environments

On the left hand side of the schema (figure 3.1), *tasks* are placed. In all kind of interactive computing systems, tasks play an important role, and often are the motive for interacting with the system.

To classify the enormous task-space which applies to 3D environments, Esposito [Esposito, 1996] defines five common task spaces:

- Navigation and locomotion
- Object query (such as selection)
- Object manipulation
- Object creation and modification
- Application environment query and modification

Another widely used classification is based upon Esposito's identification. Gabbard and Hix [Gabbard and Hix, 1997] reduce the task-space to three groups:

- Navigation and locomotion
- Object selection
- Object manipulation, modification and querying

In this view, all querying and modification of environment variables (menus, widgets, etc.) is seen as object interaction.

Other classifications can be found in Bowman's work [Bowman et al., 2005]. Bowman identifies 'selection and manipulation' as one group, 'travel and wayfinding' as a second, and finally 'system control' as a separate group.

In the next section we will first shortly elaborate on the devices which can be used in multimodal environments. Next, in section 3.4, we elaborate on some of the most common metaphors, based upon Gabbard's classification.

3.3 Devices for Multimodal Interaction

As the interaction device is the final connection between the task on one side and the physical world on the other side, devices play an important role in designing adequate multimodal interfaces. Because devices are limited to their

technical bounds, going along with only a very limited aspect of the human senses, they often are a bottleneck.

Most theories on multimodality don't take the device explicitly into account. Only Coutaz et al. [Coutaz et al., 1995] integrate devices in their theories. They define the device, together with the interaction language[1] as an interaction technique. Analogous to the CARE properties, they also define the equivalence and the assignment of a device. Respectively more devices can be used within the same language, or a device is assigned to a language. E.g. keyboard and microphone are equivalent for natural language, while the mouse is assigned to direct manipulation. In the same way device redundancy is established when a user can spell a word, while simultaneously typing the letters.

It is true that the chosen interaction technique often requires a given modality and hence limits the possible devices. However, the same interaction technique still can be applicable by switching to another device, which for instance may be more suitable in a specific environment. Therefore we prefer to say that an interaction technique *requires* certain modalities and devices (figure 3.1), rather than saying that an interaction technique *is* the combination of a modality and a device.

It is beyond the scope of this thesis to give a comprehensive overview of all possible input and output devices currently used in 3D environments. Therefore, we refer the interested reader to Bowman et al. [Bowman et al., 2005]. In table 3.1 a limited list of the most commonly used devices, classified by their medium is shown.

3.4 Metaphors in 3D

3.4.1 Metaphors in General

Definition 3.1 ***Metaphors*** *explicitly mimic concepts that are already known by the user in another context, in order to transfer this knowledge to the new task in the new context.*

[1] An *interaction language* is a language operated by the user or the system to exchange information. A language defines the set of all possible well-formed expressions, i.e. pseudo natural language or direct manipulation language [Coutaz et al., 1995]. In many aspects, this corresponds to our definition of a *modality*.

Table 3.1: List of common devices in multimodal interaction

Input		
Medium	**Device**	**Examples**
Visual	Camera	
Aural	Microphones	
Kinesthetic	Mice and 3D Mice Keyboards Mechanical trackers Gloves Magnetic trackers Optical Trackers	 PHANToM, BOOM, Microscribe,... Fastrak, Flock of Birds,...
Output		
Medium	**Device**	**Examples**
Visual	Monitors Head Mounted Displays Projection Screens	
Aural	Speakers and Headphones	
Haptic	Force Feedback devices Vibrotactile Actuators Inflatable gloves	PHANToM, Delta, Omega, HapticMaster

A well known metaphor is the standard *desktop metaphor* we know from our personal computer. Initially, the visual representation on the screen is represented as a normal desktop as in an office, completely with files, folders and a trash. The metaphor transfers the knowledge people have from their 'real' office, to the computer environment.

It is important, however to know that two constraints exist on the usefulness of a metaphor [Ware and Osborne, 1990]. First of all, a good metaphor must fit the user's previous knowledge in order to be able to establish the desired transfer. It has little sense to provide a car driving metaphor if the user doesn't know how to operate a car. Secondly, the metaphor must fit the task and the physical constraints it places on the interface. Indeed, a metaphor makes some actions easy and other actions difficult to do, and clearly this must match to the particular situation of the task.

As our every-day interaction with the physical world is multimodal, metaphors

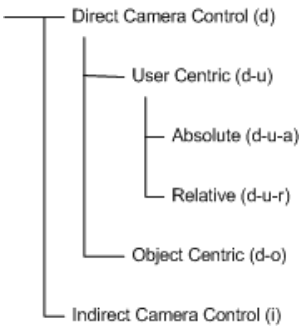


Figure 3.2: Taxonomy of Camera Metaphors [De Boeck et al., 2005a]

can be multimodal as well: direct manipulation, gestures, or speech are often used as input. Feedback is mostly given via the graphical medium, but audio feedback is frequently used as well. Haptic feedback is gaining importance in recent metaphors, as it is one of the senses users heavily rely upon in their daily life.

In the next three sections, we will go in detail to some interesting metaphors used to interact with 3D environment. Again, for a more extensive enumeration, we refer to the work of Bowman et al. [Bowman et al., 2005]. According to the classification of Gabbard and Hix, we respectively will elaborate on navigation tasks, object selection tasks and object manipulation tasks.

3.4.2 Metaphors for Navigation Tasks

Navigation is undoubtedly the most basic interaction task in a 3D environment, as the user mostly wants to *explore* the virtual world. According to Bowman [Bowman et al., 1997] the navigation task consists of two phases: a cognitive component called *wayfinding* and a locomotory named *viewpoint motion control*. In the wayfinding phase the user plans how he can reach the desired location, based upon an internal mental map of the environment together with several (mostly visual) cues from the environment. Viewpoint motion control, or *travel*, on the other hand is the physical component used to move one's viewpoint between different locations in the environment. Both phases are separate processes, although they can have an influence on each other: a mentally difficult metaphor to control the travel, leaves less time for the user to thoroughly complete the wayfinding phase, and thus degrades the entire process.

Navigation metaphors in 2D applications are often restricted to *'scroll bars'* or the well known *'hand cursor'* that grabs the canvas to move it around. When navigating in 3D space, 6 degrees of freedom (6DoF) are possible. It may be clear that we need to overcome several problems in order to provide an intuitive metaphor for 3D navigation. First of all, standard 2D input devices are not always preferable to control all degrees of freedom. It is also known that disorientation of the user will occur more easily when providing more degrees of freedom. The metaphors described in the section below will address some aspects of these problems. The camera metaphors are described according the taxonomy depicted in figure 3.2 [De Boeck et al., 2005a]. Direct camera control metaphors (d) allow the camera to be directly controlled by the user. With indirect camera control metaphors (i), the camera is controlled activating a single command that moves the camera. Direct Camera control can be split-up in object-centric (d-o) and user-centric (d-u) metaphors. Object centered metaphors allow the user to easily explore a single object, while user-centric metaphors are more suitable for scene exploration. User-centric metaphors, at their turn, can be absolute (d-u-a), relative (d-u-r) or both (d-u-a/r). In an absolute user-centric technique, a certain position of the input device corresponds to a dedicated position of the camera, while relative techniques are controlled by indicating in which direction the camera will travel.

In the following paragraphs, we will enumerate some existing camera metaphors; each metaphor will be classified within the former taxonomy (see also table 3.2).

Direct Camera Control Metaphors

In this category we find metaphors in which the user directly controls the position and orientation of the viewpoint using an input device. The device can be 2DoF (like a desktop mouse), 3DoF (like a joystick) or 6DoF (like a SpaceMouse or PHANToM device).

User-Centric Camera Control

The **flying vehicle** metaphor [Ware and Osborne, 1990], as well as the haptically controlled crafts [Anderson., 1998] represent the virtual camera as mounted on a virtual vehicle. By means of an input device, the user controls the position of the vehicle by relative movements. The flying vehicle metaphor turns out to be very intuitive, and therefore it turns out to be by far the most widely used solution when the user has to move around in a limited-sized

world. It is no surprise that in literature, several variations of this metaphor can be found.

When operating a 2DoF or 3DoF input device, the other degrees of freedom are accessed via mouse-buttons, modifier keys on the keyboard, or interaction with buttons on the screen. 6DoF devices provide the user with much more possibilities, however, allowing to control all six degrees of freedom can be distracting. Therefore, the movements of the vehicle are often limited to 'walking' or 'flying' (3DoF or 5DoF), or some rotations are limited to prevent the user from moving up-side-down. The most important drawback of this metaphor is the amount of time necessary to travel between two distant locations, when navigating in huge environments.

Zeleznik [Zeleznik and Forsberg, 1999] describes **UniCam**, a camera manipulation metaphor that relies on 2D gestures with a single-button stylus or a mouse. In contrast to common camera metaphors that are controlled by 2DoF devices, this solution doesn't require any modifier keys and thus leaving those buttons for other application functionality. One drawback to this solution is the amount of gestures that users have to learn before being able to navigate intuitively.

Other navigation metaphors include all kinds of **treadmills** [Iwata, 2004]: these solutions mostly use an implementation of the flying vehicle metaphor (but mostly limited to the ground plane), in which the vehicle is driven by physical walking movements. It is clear that this is a very intuitive way of moving into the virtual world, although heavy and expensive hardware is necessary in order to create a realistic simulation. Also the limited speed of 'human walking' can be seen as a common drawback.

Gestures of the human body [Tollmar et al., 2004] (similar to UniCam) or **gaze-directed steering** [Bowman et al., 1997], both 'relative user-centric direct camera control' metaphors, can be used to 'drive' a flying vehicle. Gestures have the disadvantage that the user has to learn those (mostly) arbitrarily chosen movements. Gaze-directed steering seems to be more easily adopted by the user, and it has the advantage that viewing and steering are coupled. However, it requires much head motion and shows up to be less comfortable for the user. Moreover, when the gaze is dedicated for steering, it cannot be used for looking to other objects, which possibly degrades the overall performance in the virtual world.

The **eyeball in hand** metaphor [Ware and Osborne, 1990] provides the user with a 6DoF tracker in the hand. When navigating, the movements of the

tracker are directly coupled to the virtual camera in an absolute manner, as if the user is holding his 'eyeball in his hand'. Although this technique provides the user with a maximum of freedom, the metaphor turns out to be very distracting. The limited workspace of the user's hand also limits the scope of the navigation, which is true for all absolute user-centric metaphors.

World in miniature (WIM) [Stoackley et al., 1995] is more than just a navigation technique: it must be seen as a more general 'interaction' metaphor. From an outside viewpoint ('God-eye's view'), a small miniature model of the world is presented. Users can perform their manipulations (including camera manipulations) in the miniature representation. It allows easy and fast large-scale operations. The WIM will be handled in more detail in section 3.4.4.

Speed coupled flying with orbiting, as described in [Tan et al., 2001], can be seen as a simplification and extension of the standard flying vehicle metaphors by automatically adjusting some parameters. This solution couples the camera height and tilt to the movement speed. In addition, an orbiting function to inspect certain objects has been integrated. This interaction turns out to be efficient when larger, but relatively straight distances have to be travelled in an 'open' scene (such as the theme park the authors show). When moving in room-like scenes, the advantages fade away. This camera manipulation technique can be classified as a relative user-centric direct camera control metaphor. The orbiting function at its turn is an object-centric technique.

In this thesis, we will also contribute to the search for more intuitive navigation metaphors. In section 5.4, we describe the **camera in hand** [De Boeck et al., 2001] and the **extended camera in hand** [De Boeck and Coninx, 2002] as two camera metaphors that require a PHANToM haptic device to control the viewpoint.

Object-Centric Camera Control

The **Scene in Hand** metaphor [Ware and Osborne, 1990] provides a mapping between the movement of the central object and the input device. This technique shows its benefits when manipulating an object as it is held into the user's hand. This solution allows the user to easily 'orbit' around the object, but it turns out to be less efficient for global scene movements.

Head tracked orbital viewing [Koller et al., 1996] [Mine, 1995] is more dedicated to immersive[2] 3D worlds. When the user turns his head, those rotations are applied to a movement on the surface of a sphere around the central object. When turning his head to the left, the camera position is moved accordingly to the right. The metaphor is 'object-centric', which means that this metaphor only applies to object manipulation, and is not suitable for larger scenes.

Indirect Camera Control Metaphors

Indirect camera control techniques such as **Teleportation**-metaphors instantly bring the user to a specific place in the 3D world. Teleportation can be activated by either speech-commands, or by choosing the location from a list. However Bowman [Bowman et al., 1998] concludes that this metaphor leads to a significant disorientation of the user.

In table 3.2, we give an overview of the aforementioned camera control techniques.

[2]Immersive solutions try to 'immerse' the user as much as possible in the 3D world, for instance by applying a head-mounted display. The advantage is a 'more realistic' experience, but the main drawback is that the user is 'cut' from the real world.

Table 3.2: Overview of Camera Metaphors

	full 6DoF	Application	Other Tasks Possible	Taxonomy
Flying Vehicle (2-3DoF device)	yes	non-immersive	no	d-u-r
Flying Vehicle (6DoF device)	yes	immersive/non-imm	no	d-u-r
UniCam	no	non-immersive	no	d-u-r
Camera In Hand	yes	non-immersive	no	d-u-a/r
Treadmills	no	immersive/non-imm	no	d-u-r
Gestures	yes	immersive/non-imm	(Sel/Manip)	d-u-r
Gaze Directed	no	immersive/non-imm	no	d-u-r
Eyeball In Hand	yes	immersive/non-imm	no	d-u-a
World in Miniature	yes	immersive/non-imm	Sel/Manip	d-u-a
Speed Coupled Flying	no	non-immersive	no	d-u-r/d-o
Scene In Hand	no	immersive/non-imm	no	d-o
Head Tracked Orb Viewing	no	immersive	no	d-o
Teleportation	no	immersive/non-imm	no	i

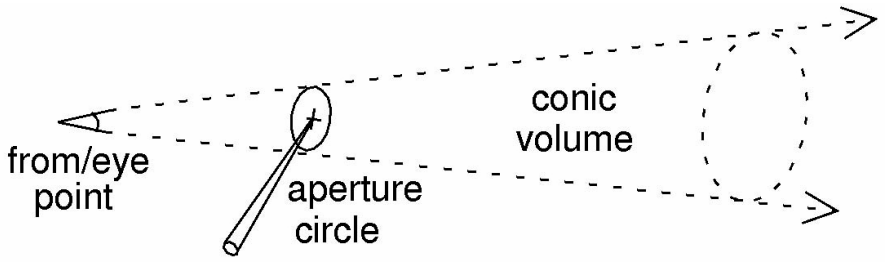


Figure 3.3: Schematic view of the Aperture Based Selection [Forsberg et al., 1996]

3.4.3 Metaphors for Object Selection Tasks

In 2D applications, the user can easily access each object on the canvas by direct manipulation. This is not true for 3D environments. Often the third dimension brings along an extra complexity in terms of completing the task in an efficient and comfortable manner. A common difficulty is the limited understanding of the depth of the world, especially when no stereo vision is available. Furthermore, it is not always possible to reach each object in the scene, due to occlusions or the limited range of the input device. Most selection metaphors try to address these common obstacles in order to make interaction more natural and powerful.

Ray-casting and **cone-casting** [Liang and Green, 1994] are by far the most popular distant selection metaphors. Attached to the user's virtual pointer there is a virtual ray or a small cone. The closest object that intersects with this ray or cone becomes selected. This metaphor allows the user to easily select objects at a distance, just by pointing at them.

The **aperture** [Forsberg et al., 1996] selection technique provides the user with an 'aperture cursor'. As can be seen from figure 3.3, this is a circle of fixed radius, aligned with the image plane. The 'selection volume' is defined as the cone between the user's eye point and the aperture cursor. This metaphor in fact improves the cone-casting by replacing the rotations of the ray by simple translations of the aperture cursor.

Other **direct manipulation** metaphors such as the 'virtual hand', 'image plane', 'Go-Go',... show their benefits for both selection and manipulation tasks. We will discuss them in detail in the next section (3.4.4).

More information about selection metaphors will be given in section 7.6. Here,

the virtual hand, ray-cast and aperture selection will be evaluated in detail.

Also **speech** [De Boeck et al., 2003b] can be used to select objects, provided that the selectable object can be named, either by a proper name or by its properties (location, size, colour, ...). At a first glance, users tend to like this interaction technique. However as the 3D world becomes more complex, it becomes more difficult (and also induces a higher mental load) to uniquely name and remember each object. Ultimately, it is also true that speech recognition is still far away from a fail-safe interaction technique, which often leads to frustration. For a detailed description of speech interaction, we refer to section 6.

Table 3.3 gives a short overview of the different selection metaphors.

Table 3.3: Overview of Selection Metaphors

	Distant action possible	Direct Manipulation	Other Tasks Possible
Ray/Cone casting	yes	yes	yes
Aperture	yes	yes	no
Virtual Hand	no	yes	yes
Image Plane	yes	yes	yes
Go-Go	yes	yes	yes
Speech	yes	no	yes

3.4.4 Metaphors for Object Manipulation Tasks

According to Poupyrev [Pouprey et al., 1998], object manipulation tasks can be divided into two classes. With the exocentric techniques (*exo*), the user is acting from outside the world, from a 'god-eye's-view'. This is in contrast to the egocentric techniques (*ego*) where the user is acting from within the world. In turn, egocentric metaphors can be divided in 'virtual hand' (*vh*) and 'virtual pointer' (*vp*) metaphors. (see fig 3.4)

Exocentric manipulation metaphors

Exocentric manipulation metaphors will execute the manipulation task from an outside viewpoint. Therefore those interaction techniques are especially usable in situations where the task is spread over relatively large distances

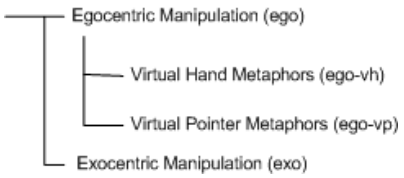


Figure 3.4: Taxonomy of Object Manipulation Metaphors

within the scene. Object manipulation tasks that require very precise interaction, such as object deformation, will be more difficult with this kind of metaphors.

The **world in miniature** (WIM) [Stoackley et al., 1995] metaphor, as described in 3.4.2, presents the user a miniature outside view of the world. This miniature can not only be used for navigation, but also for selecting or manipulating objects. This technique is especially useful when manipulations over large distances are required, but lacks accuracy due to the small scale of the miniature representation. Another drawback is the screen-space that is occupied by the WIM, although this can be solved by toggling the representation on and off.

With the **scaled-world grab** [Mine and Brooks, 1997] technique, the user can bring remote objects closer by: based on the user's arm extension, the distance to the object will be changed correspondingly. Once the world has been scaled, the interaction is similar to a virtual pointer or virtual hand interaction. According to the author, this metaphor turns out to be very intuitive.

The **voodoo dolls** [Pierce et al., 1999] metaphor is a two-handed interaction technique for immersive virtual environments. With this technique, the user dynamically creates 'dolls': transient, hand-held copies of the objects they represent. When the user holds a 'doll' in the right hand, and moves it relative to a doll in the other hand, the object represented by the right-hand doll will move relative to the object represented by the left-hand doll. This technique allows manipulation of distant objects and working at multiple scales. It takes advantage of the user's proprioceptive frame of reference between his dominant and non-dominant hand. New dolls are created using the (egocentric) image plane technique (see further in this section). Figure 3.5 shows a situation in which two 'dolls' are manipulated. The hands are added for clarity.

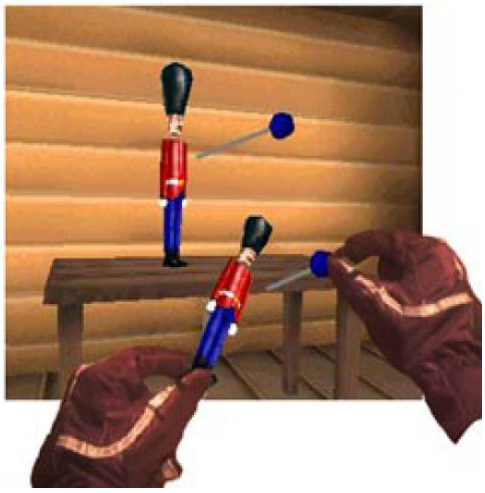


Figure 3.5: Picture of the Voodoo-doll metaphor [Pierce et al., 1999]

Egocentric manipulation metaphors

Egocentric manipulation metaphors interact with the world from a first person viewpoint. In contrast to exocentric metaphors, these solutions are generally less suitable to large-scale manipulation, but they will show their benefits as the user is directly involved in the world. Egocentric metaphors are especially suitable in relatively small-scale tasks such as object deformation, texture change, (haptic) object exploration, menu or dialog interaction and object moving and rotating.

The **virtual hand** metaphor is the most common 'direct manipulation' technique for selecting and manipulating objects. A virtual representation of the user's hand or input device is shown in the 3D scene. When the virtual representation intersects with an object, the object becomes selected. Once selected, the movements of the virtual hand are directly applied to the object in order to move, rotate or deform it. When the coupling between the physical world (hand or device) and the virtual representation works well, this interaction technique turns out to be very intuitive, since it is similar to every-day manipulation of objects. In addition, a lot of work has already been done to improve the interaction with other modalities, such as force feedback or audio feedback. The main drawback of the virtual hand metaphor, is the limited workspace of the user's limbs or the input device, which makes distant objects unreachable. This problem will be addressed in the subsequent solutions.

The **Go-Go** technique [Poupyrev et al., 1996] addresses the problem of the limited workspace by an interactively non-linear growing of the user's arm. This enlarges the user's action radius, while still acting from an egocentric point-of-view. Several variations on the Go-Go concept exist [Bowman and Hodges, 1997]. Stretch Go-Go divides the space around the user in three concentric regions. When the hand is brought into the innermost or the outermost region, the arm shrinks or grows at a constant speed. Indirect stretch Go-Go uses two buttons to activate the linear growing or shrinking.

HOMER, which stands for Hand-centered Object Manipulation Extending Ray-casting [Bowman and Hodges, 1997], and **AAAD** (Action-at-a-Distance) [Mine, 1995] both pick the object with a light ray (as with ray-casting). When the object becomes attached to the ray, the virtual hand moves to the object position. These techniques allow the user to manipulate distant objects with more accuracy and less physical effort. For the drawbacks, we can refer to the same problems we have encountered when using ray-casting (see section 3.4.3).

Ray-casting by itself is less suitable for object manipulation: once the object is attached to the ray, the user only has three degrees of freedom left, while the object is still moving on the surface of a sphere.

Image Plane interaction techniques [Pierce et al., 1997] interact on the 2D screen projections of 3D objects. The user can select, move or manipulate objects by pointing at them with a regular 2D mouse or by 'crushing' or pointing at the object with the finger. Since the 'image plane' technique is a 2D interaction technique for a 3D world, manipulating objects will not be possible with 6 degrees of freedom.

In this thesis, we also propose the **Object in Hand** metaphor [De Boeck et al., 2004a] as an egocentric manipulation metaphor using proprioceptive cues of the non-dominant hand to create a frame of reference for the dominant hand. This metaphor will be described in detail in section 7.4 and 7.5

An overview of the discussed manipulation techniques is given in table 3.4.

3.5 Summary

In this chapter we discussed multimodal interaction in 3D environments. After clarifying our vision on multimodal interaction, with 'tasks' on one hand side of the schema and physical devices on the other side, we elaborated on the different aspects of the interaction. First, we showed some existing classifications

Table 3.4: Overview of Object Manipulation Metaphors

	full 6DoF	Distant act possible	Other Tasks Possible	Taxonomy
World In Miniature	yes	yes	selec/Camera	exo
Scaled World Grab	yes	yes	selection	exo
Voodoo Dolls	yes	yes	(camera)	exo
Virtual Hand	yes	no	selection	ego-vh
Go-Go	yes	yes	selection	ego-vh
Homer/AAAD	yes	yes	selection	ego-vh
Object In Hand	yes	yes	no	ego-vh
Ray-Casting	no	yes	selection	ego-vp
Image Plane	no	yes	selection	ego-vp

for 'tasks' in a 3D environment. Next, we gave a brief overview of possible devices, classified by the medium they support. Finally, as adequate interaction techniques are necessary, the remainder of this chapter discussed the most common metaphors that can be found in literature in order to support navigation, selection and object manipulation tasks.

Chapter 4

NiMMiT: Notation For MultiModal Interaction Techniques

Contents

4.1 Introduction

From the previous chapter, it has become clear that interaction in 3D is not an easy task for the user. The designer of a 3D user interface, hence has the important responsibility to provide suitable metaphors for the requested tasks. When developing an interactive 3D interface, the designer has a large number of possibilities: choosing, combining and adapting existing solutions or developing a custom-made solution. Next he can then combine the proposed solution with a variety of devices or different input or output media. In spite of the extensive knowledge that exists nowadays, multimodal interaction is still not fully understood, and designing the optimal interaction, is often a case of trial-and-error, as the acceptance of a metaphor or an interaction technique often depends on the concrete application setup, and the user's experience and foreknowledge. The most appropriate way to evaluate a solution is still by testing it in practice in a user experiment. However, testing in practice means that each solution must be implemented separately.

In this chapter, we show a graphical notation which makes it easy to design and adapt an IT with a minimum of coding effort. The requirements for this high-level approach is two-fold:

- allowing designers to communicate about the functionality of an IT, using an easy-to-read diagram
- allowing an application framework to interpret (an XML-based equivalent of) the diagram, so it can be used for automatic execution.

As a consequence, the notation must provide enough low-level information for a framework to automatically execute the diagrams, but it also needs to be high-level and easily readable for a designer to reason about it.

Moving the implementation of the interaction to a high-level description, as we will show in this chapter, also introduces a way to easily reuse previous solutions. It allows a designer to quickly change between solutions or easily adapt existing solutions according to the findings of the test persons, resulting in a shortened development cycle.

As several powerful notations already exist, we introduce these notations in the next section, each with their particular strengths and weaknesses. Thereafter, we clarify the requirements we identified in order to describe user interaction in a 3D environment. Based upon the existing notations, combined with the

additional requirements we identify in section 4.3, we propose the basic elements of NiMMiT. In section 4.5 we explain how the notation can be used for the automatic execution of an interaction technique and how it can support collection of data for a user evaluation. Thereafter, we illustrate the notation by means of a simple example. More elaborated examples will be shown throughout this thesis.

4.2 Related Work

In literature, several relevant general notations can be found. Roughly spoken, they can be divided in two families: state driven and data driven methods. State driven notations, such as State Charts [Harel, 1987] and Petri-nets [Petri, 1962] are based on the formal mechanisms of finite state machines. A basic element of such a notation is a state transition, which has the general form 'when event a occurs in state A, the system transfers to state B, if condition C is true at that time'. Some state-driven notations are also extended to support some kind of data flow. Examples are Coloured Petri-nets [Jensen, 1994] and Object Petri-nets [Valk, 1998]. Other notations, such as Labview [National Instruments, 2006](a graphical programming language) or UML [Ambler, 2004] focus specifically on the data- or object flow. The basic element in these notations is some kind of 'activity', which contains data input and output. The execution of an activity is driven by the presence of valid data on the input ports.

When looking into the domain of user interaction, we find some notations, mostly based upon the aforementioned notations, but optimised for a specific purpose. Interaction Object Graphs [Carr, 1997] and ICO [Navarre et al., 2005][Palanque and Bastide, 1994], are mainly state driven notations, while InTml [Figueroa et al., 2002] and ICon [Dragicevic and Fekete, 2004],[Huot et al., 2004] are two very similar notations, using a data flow architecture. These diagrams consist of filters that perform the basic actions of the interaction. Each filter contains input and output ports. Output ports can be connected to input ports of other filters. The control flow of the diagram is directed by the data, in such that a filter is executed when it receives a legal value on all of its input ports.

In a preparatory study, we have conducted several experiments, describing existing interaction metaphors using different notations. We noticed that, while describing a technique using a state driven notation, the lack of data handling

can be very restricting, especially when automatic execution of the IT is required. The other way around, using one of the data flow based notations, very complex structures have to be designed in order to enable or disable certain parts of the interaction, which is an essential part of the description as well. If we combine this with the requirement that a notation must be both high-level for the designer and low-level for the automatic execution, we can conclude none of the aforementioned notations entirely fulfills our requirements. Therefore, we have developed NiMMiT as a notation which is especially suited to describe multimodal user interaction.

4.3 Requirements for Describing User Interaction

In this section, we first clarify our vision on interaction. We use a rather abstract approach, which is formulated as a set of requirements that serve as a guiding principle for the remainder of our work on interaction modelling.

In our opinion, as a result of our preceding experiments, a notation to describe interaction techniques must support following requirements:

- it must be *event driven*,
- *state driven*,
- *data driven*,
- and must support *hierarchical reuse*.

We will motivate the importance of these requirements in the context of interaction techniques:

4.3.1 Event Driven

Interaction techniques are inherently driven by user-initiated actions, which we define as events. Human interaction is multimodal by nature, so it can be seen as a combination of unimodal events (e.g. pointer movement, click, speech command, gesture,...). An event has the following properties:

- a source, indicating the modality and/or the device that caused the event (speech, pointing device, gesture,...),

- an identification, defining the event itself (e.g. button click, a certain speech recognition,...),
- a list of parameters, giving additional information about the event (such as the pointer position).

In many aspects, events are 'the initiators' of the user interaction.

4.3.2 State Driven

It is clear that, while interacting with a system, this system not always has to respond to all available events. Mostly, certain events must have been occurred before other events are enabled. For instance, the user first needs to click the pointer's button, before being able to drag an object. Therefore, we define an interaction technique as a finite state machine, in which each state defines to which events the system will respond. The occurrence of an event invokes some action to the system, followed by a state transition. For example, the dragging technique consists of two states. A first state responds to a click and brings us to the second state. The second state responds to the movements of the pointer and invokes a state transition to itself.

4.3.3 Data Driven

Limiting our vision on interaction techniques solely to a finite state machine would be too restrictive and it would violate the requirement of automatic execution. After analysing several existing interaction techniques in 3D environments, it is clear that throughout the execution of an interaction some important data flow takes place, internally; e.g. the collision between the virtual hand and an object in order to move that object. Obviously, certain data must be passed on as a parameter between the different actions within the interaction technique. Therefore, a notation to describe interaction techniques should also support data flow.

4.3.4 Hierarchical Reuse

Certain subtasks of interaction techniques recur rather frequently. The selection of objects is an example of a very common component. When modelling a new interaction technique, the designer should be able to reuse descriptions

that were created earlier, so recurring components do not have to be modelled repeatedly. In other words, the notation should support a hierarchical build-up, so that an existing diagram of an interaction technique can be reused as a subtask of a new description. Using hierarchical building blocks contributes greatly to a more efficient development.

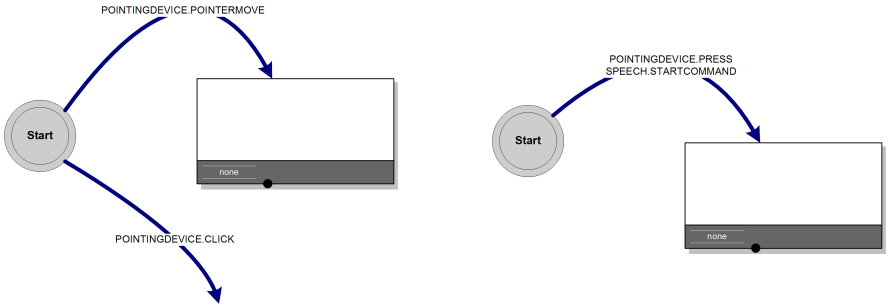
4.4 Notation Primitives

In this section, we present NiMMiT [Vanacken et al., 2006a], our notation to describe interaction techniques, based on the aforementioned considerations. This solution supports a hierarchical build-up and merges the event driven, state driven and data flow approach, in order to overcome their individual shortcomings. It results in a diagram which is easy to read for a designer while still containing the necessary data to allow automatic execution of the diagram. As NiMMiT is intended to be used by designers with a minimum of programming experience, it will become clear that, as a tradeoff, some structures intentionally are kept very simple (error handling, variables, arrays,...).

In the next section, we first explain the syntax and semantics of NiMMiT. Then, we elaborate on the automatic execution requirement, and show how the system can be used to gather user information in a usability experiment. Finally we illustrate the notation by means of a simple example.

4.4.1 States, Events and Task Chains

When activated, an interaction technique starts in its initial *state*, the start state, which simply waits for recognised *events*. Whenever a single event (e.g. a click), or a combination of multiple events (e.g. a pointer move and a spoken sentence) occurs, a task chain is fired. The dark arrows in figure 4.1(a) represent the firing of a task chain, the label shows the triggering events. When two or more labels are on the same arrow, the events must occur together (AND-operation, 4.1(b)) in order to fire the task chain. Multiple arrows indicate an OR-relation between the events. Each event is named by its source (speech, pointing device, menu, ...) and the name of the event. Some events that occur nearly continuously, such as a 'pointer move' or an 'idle' even, have a lower priority than all other events. Since events are abstracted from any physical implementation (as will be shown in section 10.2, a NiMMiT diagram is also device independent.

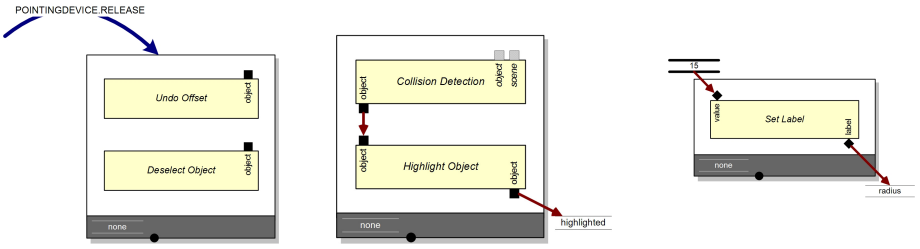


(a) Firing of a task chain

(b) Responding to a combination of events

Figure 4.1: States and events

A task chain is a linear succession of atomic tasks. Figure 4.2(a) shows a task chain (big white rectangle with grey border) containing two tasks (small shaded rectangles). The next task in the chain is executed if and only if the previous task has been completed successfully. The execution of a task often aims to change the application's internal state. The most common tasks will be predefined by the system. Some examples of possible predefined tasks in our context are 'collision detection', 'selecting', 'moving' and 'deleting' an object. Clearly not all tasks can be predefined. Therefore, we provide the possibility for the designer to add custom tasks by means of scripting.



(a) A task chain

(b) Data flow from one task to another

(c) Assigning a constant value to a label

Figure 4.2: Task chains, tasks and labels

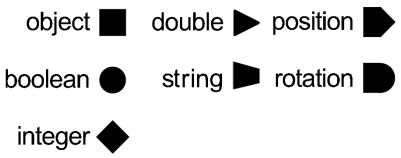


Figure 4.3: Data types and their shape

4.4.2 Preconditions, Tasks, Parameters and Labels

Each atomic task may contain a number of *input ports*, which are either required (black colour) or optional (grey colour). The tasks in figure 4.2(a) each have one required input port, presented by the small black squares at the top side of the task, named 'object'. A task may also contain a *precondition*. This is a boolean expression which indicates whether the task can be executed or not. After its precondition is fulfilled and all the required inputs are available, a task can be executed. If, for some reason, a task cannot be executed or an error occurs while executing a task, the task chain is cancelled and the interaction technique returns to the state that activated the chain.

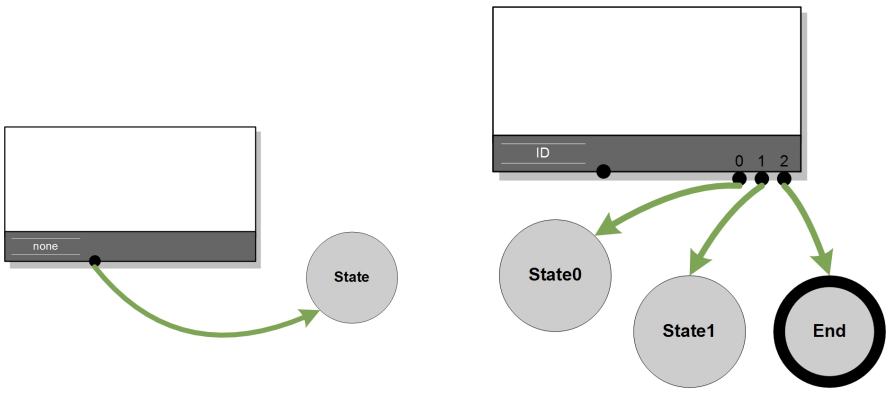
A task can produce output and pass it on via its *output ports*. These ports are similar to the input ports, but are situated at the bottom side of a task. An arrow (figure 4.2(b)) connects the output port of the first task to the input port of the next task. To prevent mistakes, input and output ports are of a certain data type (visualised by their shape as depicted in figure 4.3). Only ports of the same type can be connected to each other. All types are defined as arrays, which means that single variables are implicitly seen as one-element arrays.

Labels are high-level variables, which are known throughout the entire interaction technique. They can be connected to the output of a task in order to store values and reuse them elsewhere as input. In figure 4.2(b), the output of the second task is stored in the label 'highlighted'. This allows the designer to transfer data between different task chains in an orderly fashion. The notation also defines *constants* which are similar to labels, except for the fact that they are immutable and hence can only be connected to input ports. Figure 4.2(c) illustrates how a constant value can be assigned to a label using a predefined task.

4.4.3 State Transitions and Conditional State Transitions

After a task chain has been successfully executed, a state transition takes place. The grey arrow in figure 4.4(a) represents a state transition. The interaction technique goes either to a new state or back to the current state (a loop). In a new state, the diagram may respond to another set of events.

Moreover, conditional state transitions are possible: a task chain can have multiple state transitions associated with it. The value of the chain's label indicates which transition should be executed. Figure 4.4(b) shows a task chain with a label 'ID' and three possible state transitions. If the value of the label is zero, the leftmost transition is executed, if its value is one, the system ends up in 'State1', and if the value is two, the interaction technique jumps to the end state. A chain label can only be an integer or a boolean. This construction allows the designer to define state transitions, which are dependent on the result of a task.



(a) State transition (b) Conditional state transition

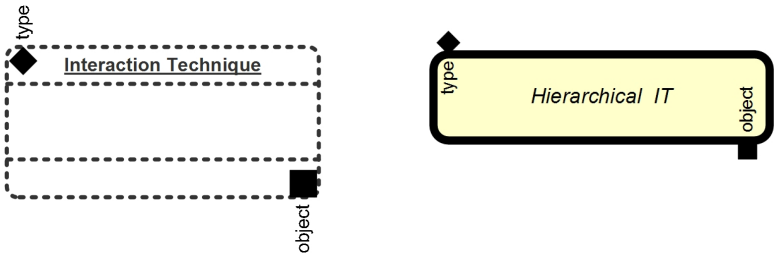
Figure 4.4: State transition and conditional state transition

4.4.4 Multiple Concurrent Diagrams and Hierarchical Use

An interaction technique itself has the same interface as an atomic task in a task chain in terms of preconditions and input and output ports. This means it has optional and required input parameters, as well as output parameters and a precondition (figure 4.5(a)). The input and output ports of an IT can be connected internally to a label. As soon as the label contains a legal value,

this value is available on the output of the IT, so that any result within the diagram be can immediately reflected outside. This feature can be used as a synchronisation mechanism between more concurrently running IT's.

The analogue interface of interaction techniques and atomic tasks opens up the possibility of hierarchical use. Indeed, each interaction technique can be used as a task in a task chain. When a hierarchical interaction technique is activated, the current diagram is temporarily suspended and saved on a stack, waiting for the newly started interaction technique to finish. Hierarchical interaction techniques are depicted as shown in figure 4.5(b)



(a) Overall interaction technique

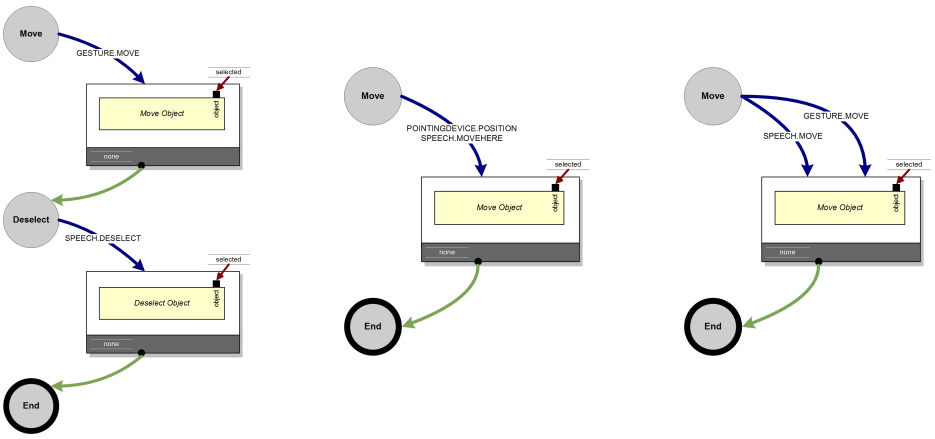
(b) Hierarchical use of an interaction technique

Figure 4.5: Overall interaction technique and an interaction technique hierarchically reused

4.4.5 NiMMiT and Multimodality

As formulated in the previous paragraphs, the notation is designed to support multimodal interaction. In the previous sections, we explained how direct manipulation, gestures and menu commands can cooperate by means of events, but a more advanced multimodal interaction is also possible. Based on the idea that a multimodal interaction is caused by several unimodal events, the notation gives the opportunity to support sequentially multimodal and simultaneous multimodal interaction, as well as the CARE properties 2.2.3.

- Sequential multimodality (or some cases of assignments and complementarities) can be implemented by defining subsequent states that respond to events coming from different sources. First, in figure 4.6(a), an object is moved via a gesture. In the next state, it is deselected by speech.



(a) Assignment, Sequential Multimodality

(b) Complementary, Simultaneous Multimodality

(c) Equivalence

Figure 4.6: Multimodal support within NiMMiT

- Simultaneous multimodality (or some other cases of complementary modalities), as well as redundancy is supported by using the AND-operator between the affecting events: the object is moved by a movement of the virtual pointer, together with a speech command as shown in figure 4.6(b).
- Equivalent modalities are carried out using the OR-operator. Figure 4.6(c) illustrates a move command that can be achieved by either a speech-command or a gesture.

4.5 Automatic Execution of a Diagram

One of the key requirements for the development of NiMMiT is that the notation should be suitable for automatic execution of the interaction technique. Therefore, a process as depicted in figure 4.7 is applied. First, the diagram is translated and saved to its XML equivalent. That XML-file, uniquely describing the diagram is then loaded and interpreted by the application. A general 'NiMMiT interpreter' is integrated in the application, and manages the automatic execution. Events are captured by the application and sent to

the interpreter. For the support of the advanced multimodality as described in 4.4.5, we have chosen to adopt the 'Melting Pot' principle of Nigay et al [Nigay and Coutaz, 1995]. A more detailed explanation on the technical aspects of this interpreter can be found in section 10.3.

Typically, a task within a task chain manipulates the internal state of the application, therefore, tasks must be built to fit the specific application in which the NiMMiT interpreter is integrated. For the application framework which is used throughout this thesis, most common tasks (such as selecting, moving, deleting, etc) have been predefined, and can be used as building blocks. Very specific tasks can be created either using Lua-script [Lerusalimschy et al., 1996][LUA Scripting, 2005], or by writing custom code in C++.

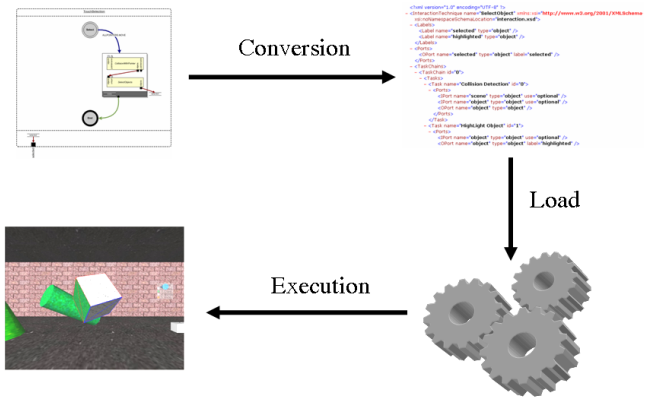


Figure 4.7: Execution of a NiMMiT Diagram

4.6 Adding Support for Usability Evaluation

In the previous sections, we have explained the basic building blocks of NiMMiT and we have shown how a diagram can be used for the automatic execution of an IT. NiMMiT allows a designer to quickly switch between possible interaction techniques, or easily adapt them. However the evaluation phase of the proposed solution still requires too much ad-hoc programming code. Hence NiMMiT has been extended to support automatic collection of data in a usability test. Clearly, adding automation to usability evaluation will have many potential benefits, such as time efficiency and cost reduction [Ivory and Hearst, 2001]. In this section, we will shortly explain how we extend NiMMiT

with three new primitives: probes, filters and listeners, in order to add support for usability evaluation. A more detailed description can be found in [Coninx et al., 2006].

4.6.1 Probes

A probe can be seen as a measurement tool that is connected at a certain place in a NiMMiT diagram, like an electrician placing a voltmeter on an electric circuit. Probes can be placed at different places in the diagram: at a state, a task chain, a task or at an input/output port of a task. An example is given in figure 4.8, in which a probe is connected to the 'Select'-state. The probe returns relevant data about the place where it is connected to, in a structured way:

State probes contain all events that occur while the state is active.

Task Chain probes contain the activation event(s) of the task chain, its status (executed, interrupted or failed), and the value of the label indicating the correct state transition.

Task probes indicate whether or not the execution of the task succeeded.

Port probes contain the value of the port to which they are connected.

Each loop, the data of all probes of the diagram is evaluated and returned. If a probe is connected to a place which was not active in the current phase of the interaction, it returns empty. In this way, NiMMiT's probes are a useful tool to debug an interaction technique. For instance, by placing a probe on all states of a diagram, one can verify the correct order of the states or check for the events that are recognised. By placing a probe on an output port of a task, the output of that task can be verified. This can lead to a significant reduction of the time necessary to find logical errors in a diagram.

4.6.2 Filters

In order to collect data for a formal evaluation of an interaction technique, the output of a probe is not always directly suitable. Therefore, we have defined the concept of *filters*. A filter can be seen as a meta-probe: a probe which listens to the values of one or more probes. As filters are probes themselves,

filters can be connected to other filters as well. A filter can rearrange or summarise the data from the probes it is connected to, but it can also just wait until legal data arrives for the first time, and then start, stop or pause an internal timer. The latter approach can be used for measuring the time spent between two states of the interaction. Although the output necessary for a user experiment can be versatile, very often the same patterns return, such as summarising a distance, counting the elapsed time or logging success or not. For these patterns, NiMMiT contains a standard set of commonly used filters such as, but not restricted to, (conditional) counting, distance measuring and time measuring. Of course, experienced users can still develop custom filters according to their special needs. In the near future we plan to extend our approach in such a way that the filters can be implemented through the scripting language also used for writing 'custom tasks'. As filters can be connected to several probes, even across diagrams, they are not visualised in a NiMMiT diagram.

4.6.3 Listeners

Filters and probes do not provide any output; they only collect and structure data. By connecting a listener to a probe or a filter, the output can be redirected to the desired output medium. By default, there are listeners that can write data directly to a file, to a text window, or even send it onto the network to an external computer which can be dedicated to handle, store or visualise the collected data. As with the filters, experienced developers can write their own listeners, if necessary. Because listeners are connected to filters, and filters have no representation in a NiMMiT diagram, they have no representation in the NiMMiT diagram, either.

4.7 Example

In order to illustrate the notation using a practical example, figure 4.8 shows a simple NiMMiT diagram, describing a click-selection in 3D. More complicated examples will be discussed in the later chapters of this thesis.

The NiMMiT diagram, shown in the picture, begins in the start-state, named 'Select'. When one of the pointers moves, the event invokes the right-hand task chain. Here, in a first task, all the highlighted objects are reset, and the empty list is stored in the label 'highlighted'. Although at first sight the output of

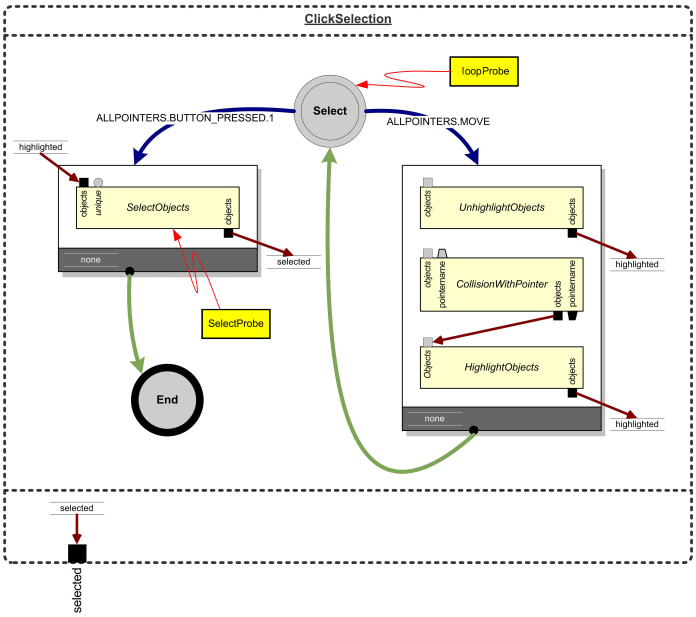


Figure 4.8: NiMMiT Diagram of a Click Selection Interaction.

an empty label appears to be a bit strange, the task chain is built in this way because the predefined tasks are designed to be more generic. In this case, the 'UnhighlightObjects' task can receive an optional parameter to unhighlight just *some* of the highlighted objects, and returns a list of the objects that are still selected. In the second task, collision with the pointer and the entire 3D scene is calculated, and the output objects are sent to the 'HighlightObjects' task, which highlights the passed objects. The result is again stored in the label 'highlighted'. After the last task of this task chain successfully finishes, the schema moves on to the next state, which in this case is the 'Select'-state again. This loop is repeated each time one of the pointers is moved.

As soon as one of the buttons, connected to the pointing devices, is pressed, the left-hand task chain is invoked. Here the highlighted objects become selected, and the selected objects are put into the 'selected' label. This label is sent to the output port of the interaction technique, ready to be used in other subsequent interactions. However, if no object is highlighted at the time of the button press, the execution of the task chain will fail because the 'SelectObjects' task requires an object as input parameter.

The example also contains two probes. One state-probe called 'loopProbe'.

Each loop, this probe returns the list of the current recognised events. The second probe 'SelectProbe' returns no value as long as the right-hand task chain is executed. As soon as the user presses the button, this probe returns whether or not the task it is attached to has been successfully completed.

The data of the probes can be redirected to a file or a network computer using a 'Listener', providing a tool to the designer to debug the IT. In order to collect data for a user experiment, e.g. a standard 'Timer filter' can be used. Connected to the two probes, this filter starts counting as soon as the 'loopProbe' generates its first output, and stops the timer when the 'SelectProbe' creates its first output. The output of the filter is the time elapsed between the first and the second trigger. Using an appropriate listener, the output can be sent to any source for statistical processing.

4.8 Summary

As the implementation of an interaction technique is often a very time consuming process, in this chapter, we proposed NiMMiT, a Notation for Multimodal Interaction Techniques. This approach allows a designer to describe the interaction technique rather than implementing it. In a first section we motivated our approach, and showed why we preferred to develop a new solution, based upon existing notation. We also explained the syntax and semantics of NiMMiT, we showed how it can be used for automatic execution of a diagram and described an extension to use NiMMiT for the collection of user data in a usability experiment of the diagram. We ended this chapter by a simple example to show our notation in practice.

In our current research, outside of the scope of this thesis, we are re-assessing NiMMiT. For a more detailed report on a comparison between NiMMiT and some data-flow notations we refer the interested reader to [De Boeck et al., 2006c].

Part II of this thesis will show the experiments we conducted towards an intuitive multimodal interaction technique. When necessary, we will use NiMMiT in order to explain the more complex interaction.

Part II

Multimodal Interfaces

Chapter 5

Haptic Interaction

Contents

5.1 Introduction

As we have seen in chapter 2, multimodality is one of the possible solutions to make interaction in 3D environments more intuitive and comfortable. Haptic feedback is one of those modalities that can be used to achieve this goal. In this chapter, we will first define what we mean by haptic feedback, subsequently we elaborate on the benefits and some existing solutions which are described in literature. In section 5.4 we explain our approach to use a force feedback device for navigation purposes.

Table 5.1: Definitions of Haptic Feedback [Oakley et al., 2000]

Term	**Definition**
Haptic	General term: relating to the sense of touch.
Proprioceptive	Relating to sensory information about the state of the body
Vestibular	Pertaining the perception of the position and accelerating of the head.
Kinesthetic	Meaning the feeling of motion, relating to sensation originating in muscles, tendons and joints.
Cutaneous	Pertaining to the skin as a sense organ, including the sense of pressure, temperature and pain.
Tactile	Pertaining to the cutaneous sense, but more focussed on the sense of pressure in particular.
Force Feedback	Relating to the mechanical production of information sensed by the human kinesthetic system.

5.2 What is Haptic Feedback

The word '*haptic*' is derived from the Greek word 'haptein', meaning 'to touch' or 'to grasp', pertaining to the sense of touch. From our every-day life, it is clear that the sense of touch consists of several perceptions. We can define force feedback as the result of a mechanical production of a force sensed by the human kinesthetic system. Tactile feedback is pertaining to the sensation of pressure, while the cutaneous sense is more general and also takes temperature and pain into account. Proprioceptive feedback is related to sensory information about the state and position of the body and the vestibular sense refers to the head position and acceleration. Table 5.1 gives an overview of the definitions we will apply in this thesis [Oakley et al., 2000].

In this chapter, we will mainly focus on force feedback, later in this document, in chapter 7, we will integrate proprioceptive feedback in our solution.

We can also make a distinction between *active* and *passive* force feedback: according to Lindeman [Lindeman et al., 1999], passive force feedback is caused by the shape of an object held or kept by the user. Active feedback is caused by a specialised actuator that actively generates forces or vibrations.

Like all the human senses, the haptic sense has its characteristics and limitations. Srinivasan et al. [Srinivasan and Basdogan, 1997] and Burdea [Burdea,

1996] summarise interesting quantitative information with respect to these characteristics. As a background, the numbers which are of any relevance in our research, are listed below.

- Humans can detect vibrations of up to 1kHz. The highest sensitivity lies around 250Hz where amplitudes of less than 1 micron can be detected.
- The kinesthetic/proprioceptive resolution of the fingers and the wrist is about 2 degrees. The shoulder is more accurate with a resolution of about 1 degree.
- The resolution for velocity and acceleration is measured as the 'Just Noticeable Difference' (JND). For the fingertip, it is between 11% and 17% of reference values.
- The bandwidth of the motion of human limbs is dependent on the kind of motion: 1-2Hz for unexpected signals, 2-5Hz for periodic signals, up to 5Hz for learned trajectories and 10Hz for reflexes.
- The maximum controllable force exerted by the finger is between 50 and 100N, but the typical forces are in the range of 5 to 15N.

5.3 Literature

The research in haptic feedback, already started in the fifties and sixties. Sutherland [Sutherland, 1965] launched his ideas on 'the ultimate display', in which the presence of haptic feedback expects an improvement of the simulation. As a result of those ideas, the University of North Carolina started research into force feedback with the GROPE project [Batter and Brooks, 1971]. This project, started in 1967, developed a haptic display for 6DoF force fields when interacting with protein molecules. Since these early research projects, the importance of this research domain has steadily grown.

In the next paragraph, we will first show some related work illustrating the benefits of haptic feedback in terms of speed and efficiency. Next, as force feedback in particular plays an important role in this thesis, we shortly describe some devices that can be used to generate this feedback. Finally, paragraph 5.3.3 enumerates some domains in which force feedback has been successfully applied in practice.

5.3.1 Benefits of Haptic Feedback

Adding haptic feedback as an additional medium in the communication between a human and a computing system may improve the efficiency of the interaction. Several studies prove the benefits of haptics in particular situations or for particular tasks. For instance, Unger et al. [Unger et al., 2002] show that the performance of a virtual peg-in-hole task improves when force feedback is available. Although they also find that the performance is still better when putting a *real* peg in a *real* hole. Dennerlein [Dennerlein et al., 2000] measured that force feedback with a 2D mouse in a desktop environment improves the performance of steering and targeting tasks with about 52%. Arsenault et al. [Arsenault and Ware, 2000] conducted a study showing that both head tracking and force feedback improve the eye-hand coordination in a tapping task. Gaulddie [Gauldie et al., 2004] at his turn produces evidence that a PHANToM device with force feedback is a better solution for 3D tasks than a standard WIMP[1] interface with a mouse. Wu et al. describe the effects of the haptic feedback on the exploration of (2D) objects. They find that curvature of the object and the stiffness of the simulation are crucial for the recognition of a shape [Wu and Okamura, 2006].

Brewster et al. suggest that the use of 'Tactons' (structured tactile messages) can provide additional information to the user [Brewster and Brown, 2004]. Similarly, Van Erp et al. show how vibrotactile feedback can help steering on the sea or in the air [Van Erp et al., 2004]. In another domain, Sjostrom et al. [Sjöström and Rassmus-Gröhn, 1999] explain how haptic feedback can have its benefits for disabled people, such as for the blind. In the same context, Magnusson uses force feedback together with audio feedback to investigate how blind people can navigate in a 3D virtual environment [Magnusson and Rassmus-Gröhn, 2005].

5.3.2 Haptic Devices

For the research presented in this text, we use Sensable's PHANToM Premium device [Massie and Salisburg, 1994] for force feedback. As shown in figure 5.2, the PHANToM is a mechanical construction with 6 encoders, constantly measuring the x, y and z-position of the stylus, as well as the three rotations. Additionally, the PHANToM contains three DC motors generating force feedback along the x, y and z axis, resulting in 6DoF input and 3DoF

[1] Windows, Icons, Menus, Pointer Interface

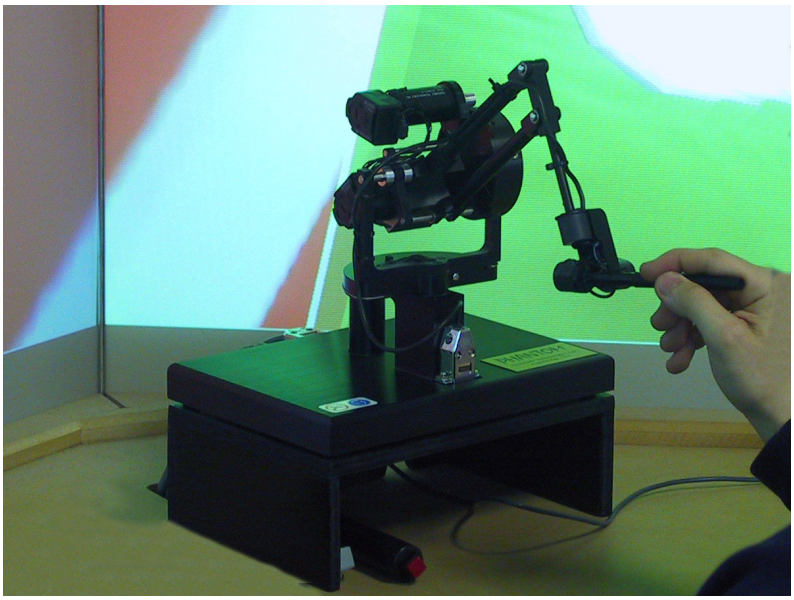


Figure 5.1: PHANToM 1.0 Force Feedback Device

output. To generate a natural feeling of force feedback, the forces produced by the motors must be updated at least 1000 times per second.

We have chosen for the PHANToM device in our research as it was the device producing the best force feedback, for years. Last years, however, other companies and research labs developed devices with similar or even better realism. Although it falls beyond the scope of this thesis to give a complete overview of all available haptic devices, some of these devices are (but are not limited to) the Delta Device [Grange et al., 2001], the Omega Device [Forcedimension, 2006], the Cubic and Freedom 6S [MPB Technologies, 2006], Haptic Master [Lammertse et al., 2002] or the SPIDAR [Murayama et al., 2004]. Those devices operate at a scale of the forearm or the wrist.

For other devices operating at the scale of the shoulder (exoskeletons) or the fingers (cybergrasp) or the fingertips (tactile arrays), we refer to the specialised literature [Burdea, 1996].

Since the research presented in this text has been carried out using the PHANToM device as the only alternative to produce force feedback, we have to take into account that this may partially influence our results. However, we believe that the results may be generalised to other force feedback devices that op-

erate at the scale of the forearm, regardless of some possible practical issues such as the placement of the device on the desktop or the device's geometrical limitations.

5.3.3 Applications of Force Feedback

From the early nighnties, haptic feedback has grown to a mature domain and has already been successfully applied in several domains. Since the usage of haptics in this thesis is mainly focused on force feedback, we will show some applications in which force feedback has been successfully applied. Stone [Stone, 2000] and Salisbury [Salisbury, 1999] both give a comprehensive overview of the different domains: going from telepresence, over training up to virtual reality. In the list below, we elaborate on some domains and enumerate some practical applications.

Force Feedback in WIMP interfaces: One of the first attempts to add force feedback to the 2D desktop metaphor, is described by Miller et al. [Miller and Zelevnik, 1998][Miller, 1998]. They apply a PHANToM device to add force feedback to the standard widgets of the X-windows desktop. Similarly, Oakley et al. [Oakley et al., 2000] prove the benefits of force feedback in a 2D environment by adding gravity wells, haptic textures and friction.

Data Visualisation: Obviously the most important output medium in the domain of data visualisation is the visual medium. However force feedback can have its merits here, as well. Visualisation of data obtained by a medical scanner has been investigated by Mor et al. [Mor et al., 1996]. Other visualisation applications include McLaughlin's visualisation of off-shore seismic data [McLaughlin and Orenstein, 1997], allowing analysts to feel data that is visually obscured by other information in the data set.

Medical Applications: A lot of 'haptic' applications can be found in the medical domain. Besides medical visualisation, rehabilitation is also an important domain. Deutsch et al. [Deutsch et al., 2001] show how a force feedback device can help patients with an ankle injury. A more recent example can be found in the Jerusalem TeleRehabilitation System [Sugarman et al., 2006] where patients can rehabilitate at their home with remote assistance from the doctor. Training of medical students is an

other important domain in which force feedback can help. Crossan et al. [Crossan et al., 2000] developed a force feedback enabled horse ovary simulation palpation, which could be used by the veterinary students. Similarly, Wang et al. [Wang et al., 2003] developed a training simulator for dentists. This system is developed to simulate two typical operations in dental surgery: probing and cutting.

Virtual Prototyping: Force Feedback also has its merits for virtual prototyping. One of the best known commercial applications, is Sensable's 'FreeForm' [Sensable Inc., 2006], but also the LEGOLAND project used force feedback to create models [Young et al., 1997].

Molecular Modeling: For chemists, the simulation of forces between an atom and a molecule or between two molecules can be of great help for the development of new molecules. In 1998, Wanger [Wanger, 1998] presented a molecular modeling application. Krenek [Krenek, 2001] focusses on the computational analysis and the realistic rendering of the molecular configuration. More recently, Lai-Yuen [Lai-Yuen and Lee, 2006] described her application for Computer Aided Molecular Design (CAMD). The tool uses force and torque feedback to assist the user to find what molecules can and cannot be assembled.

Teleoperation: In this domain, the user is operating a robot at a remote location. Force Feedback is one of the key modalities to improve the feeling of telepresence, as the operator can 'feel' the remote objects and the remotely produced forces. DLR focusses on teleoperations in space [Preusche et al., 2005], while other researchers apply telepresence for micro-assembly [Zaeh and Petzold, 2005].

5.4 Haptic Navigation in a 3D Desktop Environment

From the previous sections, it has become clear that force feedback improves the naturalness and intuitiveness of the user interaction in a variety of situations. The usage of force feedback, however, is mainly focussed on the manipulation aspect. If the 3D environment becomes larger and the generated world exceeds the physical workspace of the haptic device, navigation will be a solution (see also section 3.4.2). Very often in desktop haptic applications, a second device, dedicated to navigation, is brought in. The haptic device is

Figure 5.2: Setup with 3D mouse for navigation

then still operated by the dominant hand and it is used for pointing and manipulation operations. The second device, typically a 3D mouse, is operated by the non-dominant hand.

As mentioned in the introduction, we are heading towards a multimodal desktop environment in which force feedback and proprioception play a role in two-handed interaction techniques. As a consequence, to our opinion, the setup with a 3D mouse is not in all cases ideal. Indeed, it turns out that novice users have difficulties operating such a device with their non-dominant hand. In this section, we therefore propose and evaluate a new navigation metaphor using the PHANToM device. The proposed solution has a two-fold advantage: first of all, the navigation is performed by user's dominant hand, freeing the non-dominant hand to participate in other tasks as is shown later in this thesis. Secondly, using the PHANToM device for navigation also enables us to add additional force feedback, which is not possible with a 3D mouse.

5.4.1 Camera In Hand Metaphor

In the context of this experiment, very little can be found in literature about integrating forces in camera manipulations. General research, searching for the best navigation metaphor for a certain task have been conducted in the early 90's [Ware and Osborne, 1990]. As already mentioned in section 3.4.2, this work describes the 'flying vehicle' and 'scene in hand' as two very useful techniques. Other authors describe how to improve navigation and wayfinding in virtual environments [Satalich, 1995] by finding the best tools and cues.

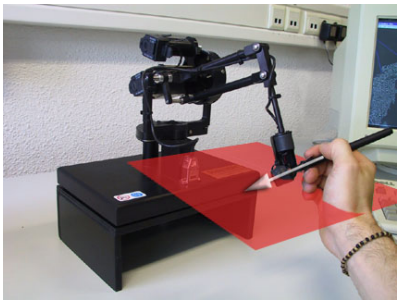
Some authors propose new interaction techniques to facilitate the user's locomotion in some particular situations. Hand-held miniatures, inspired on the 'world in miniature' are presented by Pausch et al [Pausch and Burnette, 1995]. Speed-coupled Flying [Tan et al., 2001] is proposed to navigate in large environments by making the camera's altitude dependent on the traveling speed. Finally, a usability test by Anderson provides evidence that additional force feedback in using a flying vehicle metaphor results in better performance when compared to the standard 2D navigation interface of CosmoPlayer [Anderson., 1998].

In this chapter we present the 'Camera in Hand' metaphor, which is partly inspired on the 'Eyeball in hand' navigation metaphor presented in [Ware and Osborne, 1990]. The original implementation of the 'Eyeball in hand' is to use a magnetic tracker as the virtual eyeball, which can be freely moved about the virtual scene. However, this manipulation method appeared to imply a confusing mental model in which disorientation is a common problem. In an earlier project in our research-lab, a similar navigation metaphor has been used, but controlled by a MicroScribe device [Immersion, 2006]: by moving the MicroScribe's stylus with the non-dominant hand, the virtual camera is repositioned [De Weyer et al., 2001]. By defining the viewpoint in such a manner that it matches the direction of the stylus, disorientation is strongly reduced.

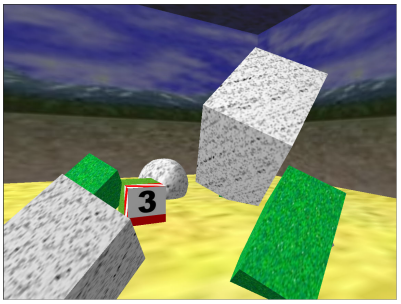
Based upon the experience we gained from this former work, the 'Camera in Hand' metaphor uses a PHANToM device for navigation operated by the user's dominant hand. The virtual camera is positioned in the direction of the PHANToM stylus, so that users can directly change their position and orientation as they are holding the 'camera in their hands'. Preliminary testing taught us to set up a horizontal virtual plane (as shown in fig. 5.3(a)) as the most useful feedback. This allows the user to easily navigate on a 'ground plane' as we are used in our daily life. When changing the viewpoint's altitude, the user has to overcome the force of the virtual plane. A formal experiment, described below, is described in order to formalise and detail the results obtained by informal testing.

Experiment

The aim of the following research activities is to formally compare this new metaphor to another existing and popular 3D navigation solution. For this comparison we have chosen to use the SpaceMouse [3Dconnexion, 2006] with



(a) PHANToM as a camera device with virtual guiding plane



(b) Virtual arena in which users have to locate the number

Figure 5.3: Haptic plane and experimental scene

a 'Flying Vehicle' metaphor.

Twenty-two volunteers with mixed experiences in virtual environments participated in the experiment. Most of the subjects were in their late twenties or early thirties, although 4 of them were above the age of 40. All subjects were right-handed and one third of the population was female.

All of the participants had to navigate in a virtual arena to locate and read a digit on a red-white coloured object (fig. 5.3(b)). The navigation test had to be performed with three conditions, ordered in a counter balanced repeated measures design. Each condition consisted of 15 navigation trials, from which the first 5 trials were considered for practicing. In the first condition we measured the performance with the SpaceMouse using a flying vehicle, secondly we tested the PHANToM device without force feedback and finally the PHANToM device with force feedback had been used. For each trial the *elapsed time* and the *total travelled distance* were logged. Finally, at the end of the test, subjects were asked to complete a comparative subjective questionnaire.

Results

Before starting to discuss the results of the test, we have to note that one of our 22 test persons had trouble in completing the tasks in *all* conditions. Since his results exceeded 12 times the standard deviation, we have omitted those values from our statistic calculations. Figure 5.4 shows us the median values of the completion times over all subjects, per trial, in each condition. This gives us a first preliminary impression of the results. The results of the

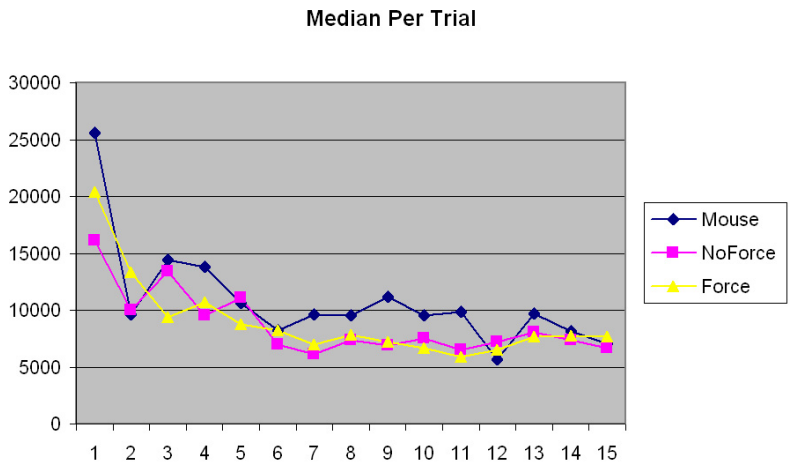


Figure 5.4: Completion Time (ms). Median values per trial

travelled distance are analogous to those of the elapsed time. We will hence limit the discussion to the elapsed time.

As can be seen from figure 5.4 and table 5.2, the average completion time in both PHANToM conditions is slightly better than when using the SpaceMouse. However, using one way ANOVA, we find a p-value of 0.12, indicating no significant difference.

Table 5.2: Averages and P-values over all subjects

(a) Average completion Times

Mouse	13333 ms
PHANToM Force	10256 ms
PHANToM NoForce	9499 ms

(b) p-values

P-Values	
Condition[Mouse-PH No]	0.1231
Condition[PH Fo - PH No]	0.8241

Because of the relative heterogeneity of our population, we have divided all subjects in four categories depending on their experience in 3D navigation: no, little, much and very much experience. Statistically, the groups with little, much and very much experience behave the same in this experiment. Therefore, in our further analysis, we consider two levels of experience: novice (users without any 3D navigation experience) and experienced (all the others).

If we look at the average completion times in table 5.3, we can see there is still no significant difference between any of the conditions in the experienced

group. However, we now notice a strong significant difference between the SpaceMouse and the PHANToM conditions among the novice users.

Table 5.3: Averages and P-values per category

(a) Average completion Times

	Novice Users	Experienced Users
Mouse	17263 ms	9760 ms
PHANToM Force	9381 ms	11060 ms
PHANToM NoForce	9007 ms	9941 ms

(b) p-values

P-Values	Novice	Expert
Condition[PH Fo - PH No]	0.0507	0.2636
Condition[Mouse-PH No]	<.0001	0.4922

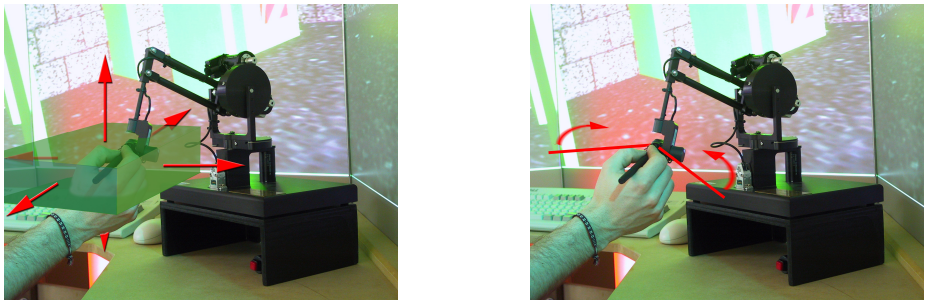
A subjective questionnaire, filled up by all subjects after the test, taught us that experienced users significantly prefer the SpaceMouse over the PHANToM. On the other hand novice users choose one of both PHANToM conditions.

Discussion

As can be seen from table 5.3 experienced users objectively do not perform differently in one or the other condition. If we look at the measurements of the novice users, we see a dramatic improvement when using the PHANToM. We can learn that the results of the novice users in one of the PHANToM conditions are similar, compared to those of the experienced category. This allows us to conclude that the proposed 'Camera In Hand' metaphor has the opportunity for the inexperienced user to perform equally to a user with more experience. However we can also conclude that the addition of force feedback,

Table 5.4: Subjective preference per category

	Novice	Expert
Mouse	2	10
NoForce	5	1
Force	3	0



(a) Bounding Box (b) Rotation Threshold

Figure 5.5: Haptic support of the Enhanced Camera In Hand

implementing a horizontal guiding plane, does not offer any advantage. The no-force condition performs even slightly better, although this difference is not significant.

As the results of the experienced users are similar in all conditions, we have to find why they collectively choose for the SpaceMouse condition. Our hypothesis is that the experienced users all spend several hours a day on a computer and all have some 3D experience playing games with mouse and keyboard. For that reason we suspect those users to have certain expectations and so feel more familiar with the SpaceMouse. In addition, some of those users report the limited workspace and tiring pose when using the PHANToM as a disadvantage.

5.4.2 Navigation Metaphor Enhancement

Based on the usability test in [Anderson., 1998], Anderson incorporated a 'craft' metaphor in the E-touch framework [Novint, 2006]. In this metaphor, the virtual camera is standing on a craft (flying vehicle metaphor). By pushing the PHANToM's stylus against the bounds of a virtual box, the craft is moving in the appropriate direction.

To step out of the limited workspace of our 'Camera in Hand Metaphor'(CiH) (reported by the subjects in the first usability test), we have combined the ideas of Anderson's craft-solution together with the 'Camera in Hand'-solution.

In the previous paragraph, we could learn that this solution allows the user to directly manipulate the camera position and hence quickly look around in the scene by pointing the stylus in any direction within its limited workspace. Now,

we limit the PHANToM's workspace with a virtual box. By pushing against the bounds of a virtual box, we stop the direct movements of the camera, and switch to a 'flying vehicle' metaphor instead (as in [Anderson., 1998]). The magnitude of the user's force exercised against the bounds of the box, gives a magnitude for the velocity of the flying vehicle (Figure 5.5(a)). The generated force feedback will help the user to distinguish between the two different modes and intuitively control the speed. Additionally, we added audio feedback when pushing against the virtual box: by pushing harder against the wall, the craft's velocity will be higher and so will the frequency and the volume of the sound of a driving vehicle.

Similarly, when the rotation of the PHANToM stylus exceeds a certain threshold (Figure 5.5(b)), the craft will automatically start rotating, while playing auditory feedback in the form of a rotating gearwheel. Most PHANToM models (except for the 6DoF) do not generate torque feedback, so the audio feedback is the only modality to give rotational feedback and therefore is supposed to be more important. Since in our daily world we mainly use only one rotation (Y-axis), the metaphor extension has been limited to this axis only. All other rotations are still kept as in the original 'Camera in Hand'-metaphor.

Assessment of the Enhancement

As the improvement of the 'Camera in Hand' metaphor is mainly focused on a better acceptance within the experienced group, the validation experiment's group consisted of only 10 experienced users. Our subjects, all right-handed males with an average age of 30, not participating in the previous test, had to perform a similar test as described in paragraph 5.4.1, while measuring the same dependent variables. Two conditions (SpaceMouse and 'Camera in Hand') had to be performed in exactly the same way as in the previous experiment, as they are used as a reference to match the results of both experiments. Additionally this experiment measured the performance with the 'Enhanced Camera in Hand'-metaphor with and without auditory feedback, as well.

Although table 5.5(a) and 5.5(b) do not show any significant difference (probably due to a smaller test-set than we had in our previous experiment), we can distinguish a trend. The Enhanced Camera in Hand Metaphor turns out to behave almost as well as the SpaceMouse and we can see an improvement of about 2 seconds in completion times between the old 'Camera in Hand" and the 'Enhanced Camera in Hand' version. There seems to be no difference between the 'Enhanced Camera in Hand' condition with or without sound.

Table 5.5: Average results of the assessment

(a) Average completion Times

Average	(ms)
SpaceMouse	8014
CiH	10333
eCiH	8274
eCiH (audio)	8302

(b) p-values using ANOVA

	p-values
SpaceMouse - eCiH	0.78
SpaceMouse - Camera In Hand	0.17
CiH - eCiH	0.25

5.4.3 Overall Discussion

In the first experiment we showed that the 'Camera in Hand'-metaphor turned out to be a solution for novice users to allow them to navigate easier than they do using a SpaceMouse with a Flying Vehicle Metaphor. On the other hand, experienced users were not convinced by our solution. A second experiment tried to address this criticism, by additionally implementing the benefits of the 'Flying Vehicle' metaphor.

Regarding the results of the second experiment, we see an improvement of almost two seconds (20%) compared to the Camera in Hand, and similar results as in the SpaceMouse condition, although we cannot show any significant improvement(table 5.5(b)).

Because the latter experiment is a continuation of the previous experiment with regard to camera and navigation metaphors, it is important that the similarities in the experimental setups are maximized, as well as that we can compare the results of these experiments with those of the former. Therefore, two of the conditions conducted in the second experiment were the same as in the first experiment (SpaceMouse and 'Camera in Hand'). Table 5.6 shows the results of both experiments. Unfortunately, it turns out that the subjects of the second experiment performed significantly 'better' than those of the first group, so that we cannot statistically compare both results.

Table 5.6: Comparison of values between two experiments

	Old Values (ms)	New Values (ms)	p-value
CiH	11059	10333	0.75
SpaceMouse	9760	8014	0.05

We can see similarities between the two result-sets in the 'Camera in Hand' condition (although they are not significantly the same), but surprisingly, we

also notice a significant difference between the values of the SpaceMouse condition. Inquiring the subjects of both result-sets about their experiences with both devices, taught us that the test persons in the recent experiment almost all had 'some' experience with the SpaceMouse, while the others did not. Since the numbers suggest that the experienced users of the last experiment performed 'better', or maybe 'more experienced' than the subjects of the first experiment, unfortunately, we cannot compare both sets plainly.

The questionnaire of the last experiment, showed an equal choice between SpaceMouse and 'Enhanced Camera in Hand' metaphors. Without any significance in the objective results, this allows us to draw a cautious conclusion that the enhancement of the 'Camera in Hand' metaphor turns out to be an improvement for the experienced user, while the original metaphor already proved its value for the novice users. Of course, this solution will have to prove its benefits in the future (as it will be used in subsequent experiments in this thesis, as well). The results of both experiments already confirm our efforts to eliminate the second input device, allowing the user to use his second hand for other tasks.

As mentioned before, force feedback allows the user to distinguish between the two modes of the 'Enhanced Camera in Hand'. Auditory feedback has been used as additional feedback in one condition. From the measurements, objectively spoken, auditory feedback has no benefits, however 80% of the users who preferred the new metaphor, appreciated the sound. Observing the users, while performing the test, we could see that users more easily discovered the possibilities of the camera metaphor and tended to push the PHANToM less frequently in extreme positions while auditory feedback was present.

5.5 Summary

In this chapter, we first defined what we mean by haptic feedback. Next we discussed the benefits and some application of force feedback, as they can be found in literature. In section 5.4 we proposed our solution to control the virtual camera position, using the 'Camera In Hand' metaphor and the 'Enhanced Camera In hand' metaphor. From a first user experiment, we could learn that the first metaphor was especially suitable as a solution for novice users to navigate through a 3D environment. Experienced users mainly complained about the limited workspace. A second experiment addressed the objections of the first, however, we could not prove a significance. Unfortunately, the subjects

in our second experiment performed better in general than the users in the first experiment, so that we could not correlate both results.

Since the haptically supported navigation, as proposed in this chapter, has shown its benefits in the described user experiment, we will use it in the next proof of concept applications throughout this text.

As a sequel of the work presented in this section, we recently presented a haptically supported steering metaphor to control an agent in a virtual environment. Roughly spoken, by adapting the spring constant of the haptic bounds, the user receives a magnitude for the travelling resistance the character experiences. When travelling uphill, the spring constant is increased, resulting in a higher resistance. Similarly, while travelling downhill, the constant is reduced. In the same way, collision with objects within the scene can be simulated. Since the actual work partially falls out of the scope of this thesis, we refer the interested reader to the particular publication [Jorissen et al., 2006] for a more detailed description.

In the next chapter, we will investigate the value of speech input in the interaction with a 3D environment. After that, we proceed to our solutions using bimanual input using proprioceptive feedback.

Chapter 6

Speech Interaction

Contents

6.1 Introduction

In chapter 5, we explained that haptic feedback is one of the modalities that make interaction with a 3D virtual environment more intuitive and agreeable. The human haptic sense is an important modality in our daily interaction with the world. Indeed, the gestural channel (body movements) is the only channel which supports the three functions: epistemic, semiotic and ergotic, as we explained in section 2.2.5. Another modality we heavily rely on in our daily existence, are spoken linguistic messages [section 2.2.2: (+Li, -An, -Ar, +Dyn, Au)], or in other words, the human natural language. Anderson states that humans are *compulsive talkers* [Andersen, 2000], suggesting that everything in our society must be 'verbalisable' in order to use or to understand it.

In this section, we investigate how natural language can cooperate in a 3D virtual environment as an input modality together with force feedback.

6.2 Related Work

As technology advances, the use of speech interfaces has also grown over the last years. Dependent on the application, speech input can, in some cases, be seen as an alternative for direct manipulation. Some successful applications of speech interfaces can be found in [Dix et al., 1997]: going from interfaces for disabled people, to voice input systems for airplane engineers occupied in a technical inspection. Other recent examples are applied in hand-held devices or in phone interfaces.

Speech interfaces have the potential to free the user's hand by controlling the environment by voice alone. It turns out to be a useful modality when

- the user's hands or eyes are involved in another task (such as driving),
- the user is disabled (either visually or locomotory impaired),
- mobility is required (such as airplane engineers sending the results of technical inspection to a central system).

Although natural language at itself is a very natural modality for the communication between humans, the use of speech interfaces does not necessarily appear to be ideal when applying it in a human-computer dialog, because of several issues. Some are technological and may be solved in the future, other issues however are typically for the nature of speech, and may inherently compromise the acceptance by the users. Some of the most important issues are summarised below.

- Computers still do not 'understand' human language, which implies that we 'pronounce commands' instead of just 'talk'.
- Speech is a very error-prone modality [Cohen, 1992]: background noise or other nearby voices may adversely influence the speech recognition.
- The speed and the pronunciation of a spoken phrase can differ between dialects or between persons. Hence, for optimal recognition a training period per user is required.

- The heavy mental load to remember and formulate commands, together with the limited capacity of the short time memory [Miller, 1956] make it difficult to pronounce (long) commands without hesitations.
- Speech turns out to be a very slow modality concerning the amount of data which can be transferred per unit of time.
- Task scheduling or problem solving and speech cannot operate in parallel in our brain [Dix et al., 1997]. Alternatively, task scheduling and locomotory movements (e.g. in direct manipulation) do operate in parallel.

Given all those issues of speech interfaces, it is not realistic to aim for a computer interface which is purely speech driven, except for some very specified applications. Moreover, Oviatt disproves some common myths and overestimations of speech interfaces [Oviatt, 1999]. As Cohen [Cohen, 1992] shows the complementarity of natural language and direct manipulation, it is more reasonable to see natural language as an addition to the existing interfaces, either as a redundant, complement or equivalent of a direct manipulation. Table 6.1 shows a brief overview of the complementarity between natural language and direct manipulation (adapted from [Cohen, 1992]). More strengths and weaknesses can be found in the original document.

As natural language and direct manipulation are clearly complementary, we investigate in this chapter 'if' and 'how' speech can help to make interaction in a 3D virtual environment more intuitive. We investigate if speech input can solve some of the problems we have mentioned in chapter 1, such as the lack of a reliable depth perception, or the limited workspace of the interaction device.

Already in the early eighties, Bolt showed that speech and pointing can seamlessly cooperate. His 'Put That There' concept is often taken as a basic example [Bolt, 1980] of what is called modality complementarity [Coutaz et al., 1995] or coordinated simultaneous multimodality [Sturm et al., 2002](see section 2.2.3). Later, Nigay et al. propose their 'Melting Pot', a generic framework which solves the synchronisation problem between the different modalities [Nigay and Coutaz, 1995]. Although several experiments can be found using speech input in 3D virtual environments [McGlashan, 1995][M.Cernak and A.Sannier, 2002], not much work can be found combining speech with direct manipulation and force feedback within a 3D environment.

As a first evaluation approach, we elaborate on a multimodal interface which is dedicated to a simple scene modeling task. The proof of concept application

Table 6.1: Complementarity between natural language and direct manipulation [Cohen, 1992]

Direct Manipulation Weaknesses	**Natural Language Alternative**
• Difficult to express Quantities Negations Temporal relations	• Descriptive, easy for Quantities Negations Temporal relations
• No anaphora (e.g. pronouns)	• Easy to use anaphora
• Delayed actions are difficult	• Delayed actions are possible
• Operations on large sets of objets	• Context, anaphora
Natural Language Weaknesses	**Direct Manipulation Alternative**
• Coverage is opaque	• Consistent Look and Feel • Options are apparent
• Error prone	• Fail Safe
• Ambiguous	• Direct engagement • Acting here and now
• Difficult for navigation	• Point, act: Direct navigation

allows us to find an appropriate combination of both modalities; speech and direct manipulation. In this work we restrict the interaction to sequential multimodal interaction. The results of this research should provide us with a sound basis to progress to simultaneous multimodal applications.

In a first section, we will explain the experimental setup. We describe our hardware configuration, our interface proposal and task subset, and finally the additional head tracking. In a next section, we will give more details about the user experiment. Subsequently, the results of the experiment will be discussed in detail. Finally we will conclude 'if' and 'how' speech can enrich the multimodal interface when interacting in a 3D environment.

6.3 Experimental Setup

The experimental setup includes a PHANToM 1.0 device, mounted in our 'Personal Surround Display' (PSD) (as described in chapter 1). The large

Figure 6.1: Experimental task

size of the PSD often forces the user to make relatively large movements with the virtual pointer. Moreover, the large scaling factor between the real and the virtual movements, also makes it difficult to access objects or menu items within the 3D world. This may also motivate the addition of speech input.

6.3.1 Task with Speech and Haptics

As a proof of concept application, we built a very limited and simple 'scene modeller'. One of the tasks that can be performed by the interface is 'changing the texture of an existing object'. Therefore, it is necessary that the user can select and deselect objects and can choose from a set of textures. To investigate the appropriate blend of both modalities, in this experiment, all commands can be performed by all modalities (equivalence). Commands can be performed by direct manipulation: by picking commands from a haptic hybrid 2D/3D menu [Raymaekers and Coninx, 2001] and by touching the objects in the virtual world. The direct manipulation has been improved with realistic force feedback, giving the impression of actually touching the objects in the world.

Alternatively, all commands can be executed by using voice commands, as well. For instance, an object (e.g. the left cube) can be selected by pronouncing the phrase 'Select left object'. The speech recognition module in this experiment, uses the Microsoft Speech API and runs on the same machine that supports the force feedback simulation. The speech recognition is only available in English, which makes that our subjects are not able to communicate in their mother tongue. A synergetic combination of both interaction methods is possible too:

e.g. the menu can be activated by voice, but the commands within the menu can be selected with the haptic device, or the object can be selected by voice, while its texture is changed by picking the texture from the menu.

It is clear that this test application only implements a very limited but relevant command set of a modeling tool. Obviously, the smaller the command set, the more efficient the speech-modality will turn out. Although it is difficult to simulate the workload of a real modeling application in a first proof of concept, we have provided our command set with a sufficient number of textures and a sufficient number of alternative expressions, so a certain load of the subject's short-term memory is achieved.

6.3.2 Head Tracking

For this experiment, we have chosen to constantly place the menu at the left screen in order not to obscure too much of the 3D world. Since the application, running in the PSD, establishes a large workspace, the user has to make relatively large movements with the PHANToM device. As this can aversely affect the acceptance of choosing items from the menu, we have added additional head tracking to the system. A tracker (Polhemus Fastrak), mounted on a cap has been used to get the orientation of the user's head. This information is solely used to get the direction in which the user's head is turned; the projection transformation is not affected by the tracker's data. When enabling the menu (invoked by a speech command), the tracker's information is used to show the menu in the user's region of interest. We believe this solution can lead to a better cooperation between both modalities, as it avoids not only large cursor movements to access the menu, but it also limits the user's workload, since the menu is brought to the user and must not be sought after.

6.3.3 Experimental Task

To examine which part of the task to change an object's texture is performed by which modality, an informal usability test had been conducted. Five subjects, all male with no or little experience using speech recognition systems and force feedback, were asked to perform a predefined set of tasks on three cubes, each of which was visualised on one of the three projection screens. Before starting the experiment, each subject was allowed to practice for 5 minutes to get used to the environment and to learn the vocabulary. Next, the test consisted of simple tasks (such as selecting an object or setting a texture) and

combined tasks (such as activating the menu, selecting an object and changing the texture). During the test, each command, succeeded or failed, spoken or pointed, was manually logged. Finally, a questionnaire was presented by which the subjects could give their subjective feedback. Two hours after the first test, the same subjects were asked to perform another trial, with similar but slightly different assignments. This test had to be executed without any practicing, to minimize short-term memory influence.

6.3.4 Results and Discussion

First Trial After Practice

From the first trial, we can conclude that speech is used as the main input modality. Almost 80% of all commands were vocal. The haptic interaction is rather used as a backup when a speech command is not recognised for a couple of times. In the survey, most users reported that speech recognition works comfortable and 'surprisingly' adequate, although still 25% of the spoken commands result in no or an unexpected result. In contrast, direct manipulation has an error rate of only 9%. However, subjects report difficulties accessing the menu, which subjectively makes the haptic interaction to appear slower. We have also reported those common problems with haptic interaction in our previous work ([Raymaekers and Coninx, 2001],[Raymaekers et al., 2001b]).

Although the users have had the possibility to extensively practice the spoken command set, we also observe that long commands with more variables are more error prone. The command '*Set left object's texture to wood*' failed in nearly 78%, due to mispronunciations or hesitations. Even the phrase '*Select left object*' failed in 38%. Shorter commands, such as to change to selection mode or to show the menu, were much more reliable.

Second Trial Without Additional Practice

Two hours after the first trial, subjects were asked to perform another set of instructions, this time without a practicing session. In this second trial we clearly see the effect of the short-time memory: some speech commands are avoided, or fail more often because users hesitate or don't remember the exact wording. The effect is stronger for the longer commands. The most complicated command '*Set left object's texture to wood*' was only tried once, without success. Also the command '*Set texture to wood*' was 'remembered'

in a lot of 'variations', which resulted in a success ratio of only 17%. Hence, direct manipulation has been used more intensively. The menu command to change the object's texture has been used twice as many times as in the first experiment. From the subjective questionnaire, however, we can conclude that our subjects still prefer verbal interaction, but they report that remembering the command set is the main problem. Therefore, the menu is more often activated as a reminder[1] (20 times against 5 times in the first trial), as some kind of 'What can I say'-feature. In this context, it is understandable that the feature to show the menu where the user is looking at, has been evaluated positively.

If we take into account the individual behaviour of the subjects between both trials, we see that two users behave in the same way in both experiments. Two users incline to interact more haptically in the second experiment. Finally one user tries to use more speech commands, but falls back to direct manipulation very often, because of too many speech recognition errors.

Discussion

Although the experiment was informal, and we only measured the results of 5 subjects, we can conclude that speech is the preferred modality, although roughly 25% of the commands failed. One can argue that English as a foreign language increases the amount of errors. This is certainly true, but on the other hand, users evaluate the recognition as 'surprisingly working well' and prefer it over the direct manipulation. Haptic interaction was reported to be a valid backup when spoken commands were unsuccessful. When eliminating short-term memory effects, in a second trial, spoken commands become less accurate and users more often fall back to the 'backup' modality. This is certainly the case for longer commands with more variables. But even then, users still prefer the oral interaction.

Although this little experiment shows the enthusiasm of the subjects to use the speech modality, we can conclude from the results that the success rate for speech commands, especially for complex tasks, is rather limited. In this context, Sturm et al [Sturm et al., 2002] demonstrate that a prolonged use of an interface supporting both direct manipulation and speech, will end in a less frequent, but more efficient use of speech commands.

[1] We consider the menu is opened as a reminder when the user intensionally opens the menu and closes it back, without picking a command.

Taking this into account, we see the benefits of speech in a 3D user interface, but its importance must not be overestimated, and this is certainly true as the complexity of the task-set grows. To our opinion, the use in a 3D environment must be restricted to a few short commands, e.g. to perform a mode or a context switch whether or not combined with some information from the direct manipulation. In this context, Alan Dix' quote may be kept in mind:

> "Speech is the bicycle of user-interface design: it is great fun to use and has an important role, but it can carry only a light load" [Dix et al., 1997]

6.4 Summary

In this chapter we investigated 'if' and 'how' speech input can be used to make the interaction with a 3D environment more intuitive. We described an informal user experiment, in order to investigate how speech and direct manipulation can cooperate and coexist in a 3D environment. The proof of concept application focuses on a small but relevant task-set of a scene modeling application. In this application both modalities were equivalent for each task.

We could conclude that users prefer speech input, although the number of errors is extremely high, especially if we take the very limited task-set into account. We therefore conclude that speech can have its benefits, as a complement of direct manipulation, but its importance must not be exaggerated.

In the next section, we will show how two-handed input can be applied to our setup, in order make the 3D world more accessible and easier to use.

Chapter 7

Two-Handed Interaction

Contents

7.1 Introduction

In the previous chapters we elaborated on the improvement of the user interaction by adding new modalities, such as force feedback and speech input, to the interface. On the other hand, the optimal use of our kinesthetic apparatus within the interface can improve the interaction, as well. Indeed, most direct manipulation interfaces use a single pointer (with or without force feedback) to interact within the generated 3D world. In this chapter, we investigate how a two-handed approach can lead to a more intuitive interface. We will also show that in this context, proprioceptive feedback, which is one kind of haptic feedback (table 5.1), can be applied to further optimise the use of both hands. We start in the next section with pointing out some related work in this domain.

7.2 Related Work

In the past, as already elaborated in chapter 3, several interaction techniques have been investigated in order to make the interaction in 3D less cumbersome. Most of these (early) solutions have in common that they require only the input of one hand. For instance, in ISAAC [Mine, 1995], objects could be accessed by pointing at them ('action-at-a-distance'). This approach was expanded in CHIMP [Mine, 1996], where menu items could be selected by looking at them ('look-at menus'). And also the 'World In Miniature' (WIM) paradigm was presented to provide the user with a miniature version of the virtual environment in order to allow for large scale manipulations.

More recent research proves that bimanual interaction improves the interaction, as this is a natural means of interacting by humans. In this context, we can distinguish two types: *symmetrical bimanual interaction* and *asymmetrical bimanual interaction*[1]. Symmetrical bimanual interaction requires both hands to perform equally in a task. An example is 'typing on a keyboard', where the two hands work at a similar level of spatial and temporal detail. Another example is the common setup in a virtual environment, where the dominant hand is used for pointer manipulations and the non-dominant hand for navigation. As long as such a two-handed task is not 'embodied' (section 2.2.5), visual feedback is of an utmost importance. Balakrishnan [Balakrishnan and Hinck-

[1] In literature, asymmetrical bimanual interaction, is sometimes called cooperative bimanual interaction.

ley, 2000] shows that in a symmetric bimanual task the parallelism between both hands increases as the visual representations are closely connected.

Asymmetrical bimanual interaction plays by far a more important role. Here, both hands perform a different part of the interaction, and as the task becomes more difficult, Hinckley found that the importance of the specialisation between the dominant and the non-dominant hand increases [Hinkley et al., 1997b].

In general, in a cooperative bimanual action, the non-dominant hand creates a frame of reference for the dominant hand. Several examples can be found, such as writing on a sheet of paper (the non-dominant holds and keeps the paper in place), putting a thread through a needle (the non-dominant hand holds the needle), striking a match (the non-dominant hand holds the box), etc. Guiard [Guiard, 1987] has proposed a theoretical framework for the study of this asymmetry with the following principles:

- The dominant hand moves relative to the non-dominant hand. In other words, the non-dominant hand creates a frame of reference for the dominant hand. E.g., holding a sheet of paper while writing.
- The non-dominant hand's movements are low in spatial and temporal frequency, while the movements of the dominant hand are more precise and faster.
- The action of the non-dominant hand in the global bimanual task starts earlier than the dominant hand's movement. This is obvious since the non-dominant hand first has to create a reference frame before the other hand can start its task.

Hinkley shows that in our every-day life, interaction with both hands creates a frame of reference which is strong enough, so that it is even independent of visual feedback [Hinkley et al., 1997a]. In [Balakrishnan and Hinckley, 1999], the authors show how the match and mismatch between the input of the hand and the visual feedback influences two-handed performance. Another technique is to introduce a grounding object to support the user's hand. The combination of a grounding object, combined with both hands provides even better possibilities. Hinckley also found that uni-manual operations are more dependent on visual feedback.

The independent frame of reference brings us to another important aspect of bimanual interaction: 'proprioception' [Oakley et al., 2000]. This is the

sensory knowledge of our body to know the position of its parts, relative to each other. This haptic feedback allows us e.g. to touch the fingers of the other hand without looking, or to put something in our mouth without looking in a mirror.

A number of techniques to facilitate 3D interaction have in common that they make use of proprioception, as well. Although applied in a slightly different domain, this bimanual approach can be found in the work of Ishii, where tangible bits [Ishii and Ullmer, 1997] are used as physical handles. Those tangible user interfaces (TUI), which can be manipulated by both hands, allow the user to intuitively work with physical objects representing the computational data. Lindeman et al. [Lindeman et al., 1999][Lindeman and Templeman, 2001] use passive feedback devices such as a physical plate, held by the non-dominant hand, for widget interaction. The user's proprioceptive knowledge of the non-dominant hand with respect to the dominant hand also appears to be a valid approach to easily activate several functions within the world. In the same context, Mine [Mine and Brooks, 1997] concludes that hand held widgets, which rely on proprioceptive information are to be preferred over floating or object-bound widgets.

7.3 Bimanual Haptic Interaction

In a first approach to integrate bimanual interaction in our haptic setup, we aimed for a bimanual *haptic* setup. Here, two PHANToM devices are used: one for the dominant, and one for the non-dominant hand. When using the GHOST API [SensAble, 2001], which comes standard with the PHANToM Device, two devices can be connected to the same computer. In section 8.3, however, we show that the maximum object or scene complexity in order to meet the requirement of a 1KHz update rate is rather limited. Only objects up to 65 000 polygons can be rendered using two PHANToMs, while with only one device, objects with more than 100 000 polygons can be shown. The results are even worse when looking at the scene complexity: no 8 objects of 1024 triangles can be rendered with two PHANToMs while 64 of those objects run fine with one device connected.

These results suggest to adopt a distributed approach, in which the haptic scene is distributed among several computers, each of them driving a separate PHANToM device. Figure 7.1 illustrates the practical setup in our lab. As this approach requires quite some technical adaptations, we will elaborate on this

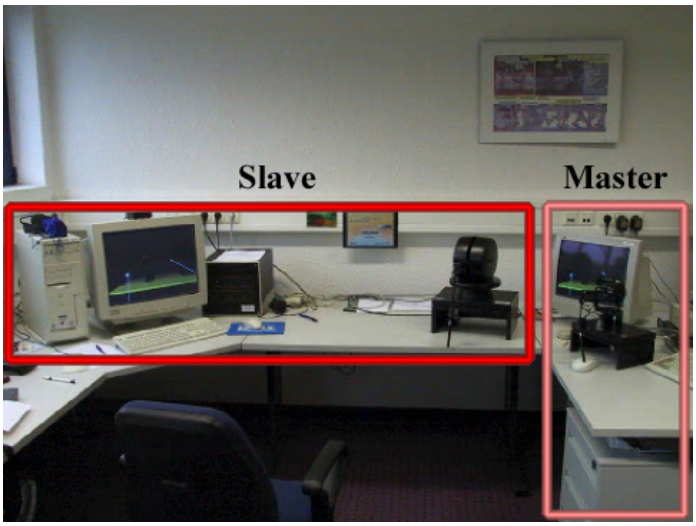


Figure 7.1: Master slave setup for bimanual interaction with two PHANToM devices

more in detail in section 9.3. In that section we will also evaluate the objective results of the implementation, such as network delay and synchronisation between the modalities (visual, audio, haptics).

After a technical evaluation, which turned out to be successful, some other practical problems rise, which make the dual PHANToM setup less suitable in our setup.

1. When mounted in the PSD, two PHANToMs cover quite some space of the projection area, giving an uncomfortable feeling for the user.

2. A PHANToM device only calculates the force in a single point in space, as if the user is manipulating the world with his finger, or (in our case using the stylus) with a pen. Two PHANToMs consequentially allow the user to manipulate the world with a pen in each hand. In most cases, this will lead to symmetric bimanual interactions, in which both hands play an equal role. However, in practice, an asymmetrical bimanual setup is preferred.

3. Finally, and probably the most important problem with this approach is the lack of proprioception. In a normal operation of the PHANToM,

there is no colocation between the physical device and the virtual representation of the cursor. This implies no specific problems, since it is known that users can quickly adapt to this mismatch. However, when two PHANToMs are applied, proprioception between the two hands will also play a role. Both PHANToMs are placed far from each other, each on one side of the body, but even if they should be placed closer together, they cannot cross each other. Proprioception between both hands, hence, is not possible in this setup, which is a problem even in a simple task such as the 'Virtual Percussionist' [De Boeck et al., 2002b].

Although the arguments listed above are not derived from any formal evaluation, they came up while integrating the setup in the development phase. We believe the setup with two haptically supported hands certainly can play an important role in some situations, but it turns out to be less suitable in our setup. Therefore, we decided to abandon this approach and look further for other solutions, which can integrate two-handed interaction together with proprioceptive cues in a more intuitive and acceptable manner.

7.4 Object In Hand for Menus

As we left the aim to provide both hands with force feedback, we have the ability to focus more on the proprioceptive feedback. This allows us to find a solution, which specifically addresses the problem of accessing objects in a 3D world. It could be observed in several of our former experiments, that accessing or touching objects or menu items in a 3D world is often a difficult task to perform.

In order to overcome this problem, we added the advantages of proprioception to our haptic interface. Instead of moving the haptic pointer to an object somewhere in the virtual space, the user's non-dominant hand brings the object to the pointer. This has the advantage that the object of interest, or a specific location within the object, has not to be sought for. Our approach adopts some principles from [Balakrishnan and Hinckley, 1999], [Hinkley et al., 1997b] and [Mine and Brooks, 1997] in such a way that we create a proprioceptive frame of reference in which the user can carry out the manipulations with the dominant hand relative to the object held in the non-dominant hand. The value of this contribution, however, is that proprioception is combined with active force feedback provided by a PHANToM device as if the user is manipulating an object held in his hand. We call this solution the 'Object In Hand' metaphor.

In a first step, as described in [De Boeck et al., 2004b] and [De Boeck et al., 2006d], we will evaluate this novel approach using the manipulation of menus only. Next, as described in [De Boeck et al., 2004a], we generalise the approach so that it is suitable for all objects, as well.

7.4.1 Haptic Widget Manipulation

In our first proof of concept application, we have used hybrid 2D/3D user interfaces [Coninx et al., 1997]. Such a user interface consists of 2D user interface elements (UI Elements) that are positioned in the virtual environment and are accessed in the same manner as the environment's objects. This has the advantage that the user can work with a user interface that consists of 2D menus and dialogs (which are known from the standard desktop metaphor), but does not have to switch to a mouse and keyboard-driven interface to access those interface elements.

We have found that, by making the UI Elements haptic, they can be used more efficiently [Raymaekers and Coninx, 2001]. But, because haptic feedback is

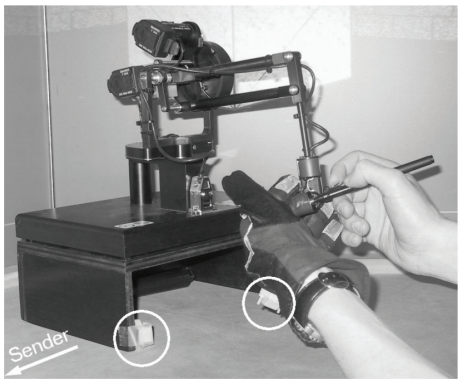


Figure 7.2: The magnetic trackers' setup

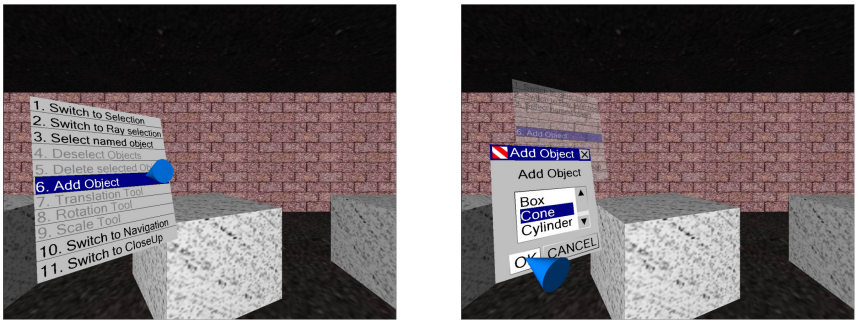
only applicable when the user hits the object, it still remains difficult, however, to exactly locate and access the element in 3D space.

Setup

In order to provide the frame of reference for the user's dominant hand, we have equipped the user's non-dominant hand with a glove onto which a magnetic tracker is mounted. Due to the need to connect the glove with a wire, it seems to be quite an intrusive device, but we believe that the near future will bring us solutions with less intrusive tracking possibilities (e.g. optical tracking), and that the results with this solution are generalisable to other similar setups.

A second tracker is mounted onto the PHANToM device's base. This gives us the opportunity to implement differential tracking and hence track the user's non-dominant hand with respect to the PHANToM device (which is operated by the user's dominant hand). As a final result, we can determine the position of the users hands with respect to each other.

The magnetic tracker on the PHANToM's base is attached in such a manner that both receivers stay between the magnetic sender and the metal of the PHANToM's construction (see circles in Figure 7.2), thus reducing the effect of the distortion in the magnetic field around the PHANToM device. Since both trackers experience a similar distortion, the combined error is less than 1mm. The effect of a possible distortion is further reduced by the fact that the haptic and visual output are kept synchronised at all times. Moreover, this differential setup has the advantage that no specific calibration for the



(a) Main menu (b) Submenu (parent faded)

Figure 7.3: Menu item calling a sub-menu

magnetic sender or positioning of the PHANToM device is required: even if the user inadvertently shifts the PHANToM device, the position of the two hands with respect to each other is still known.

Next, force feedback using the PHANToM devices improves a natural feeling of actually touching the object at that location. In this first implementation, when the user brings the non-dominant hand in the proximity of the PHANToM, a UI Element (such as a menu or a dialog) is attached to that hand and can be moved accordingly. The user's dominant hand eventually can rest on the non-dominant hand using the haptic UI-element in a PDA-like manner. In conclusion, we believe the proprioceptive approach solves the problem of interface elements being difficult to find in the virtual environment, as they can be found and accessed using proprioception.

User Interface Elements

The widget-set used in this application consists of menus, dialogs, buttons, lists, etc. Dialogs and menus can be fully described in VRIXML, an XML based description language we have designed to describe user interfaces within virtual environments [Cuppens et al., 2004]. When the non-dominant hand is brought closer, the activated UI-element can be a menu (containing commands), or a dialog (containing buttons, lists and sliders). Each button or menu-item can fire commands as known from the standard desktop user interface. These commands will be handled directly by an abstract event handler as described in section 10.2 and [De Boeck et al., 2004c]. Instead of directly firing commands, menus and dialogs can also call subdialogs and submenus. The

behaviour of either directly calling menus or sub-items is defined from within the description language and is handled by the UI element itself. When a submenu is called, the main menu will fade to the background, and the submenu will be attached to the movements of the non-dominant hand (figure 7.3). We believe this approach diminishes the risk of the user getting lost in a hierarchy of menus and dialogs.

Device Scaling

To allow the user to intuitively interact with the grabbed objects or menus, we had to take into account the different interaction scales: the haptic scene modeling in the PSD is a large-scale interaction, while the proprioceptive menu interaction works on a much smaller scale. Moving the PHANToM device over the length of a hand in normal operation causes the virtual pointer to move over a large distance in the virtual world. This is not desirable when interacting locally, since it degrades the feeling of proprioception. We have therefore chosen to scale the PHANToM device's coordinates down in such a manner that a movement over the virtual menu corresponds with a realistic movement over the user's non-dominant hand. Because such a scale would bring the PHANToM device's virtual representation to the center of the virtual world, an offset is added to the new position in order to keep the PHANToM device in the same place. The applied formula is shown in equation 7.1, where P_d is the PHANToM device's position at the time that the menu is brought in a stable position (velocity of the menu is below a certain threshold), P_p is the PHANToM device's current physical position and P_v is its new virtual position. The scaling factor is represented by α, where $0 < \alpha \leq 1$. This value depends on the size of space that the virtual pointer can move in, and the size of the menu.

$$P_v = \alpha \cdot P_p + (1 - \alpha) \cdot P_d \tag{7.1}$$

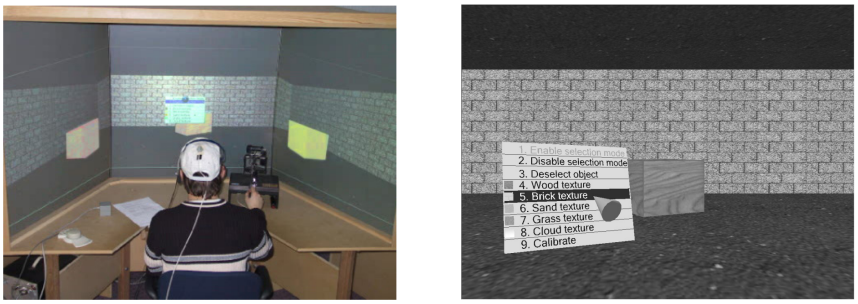
At the moment the menu is activated, P_p is equal to P_d. Thus, the virtual pointer position stays located at this position.

7.4.2 Evaluation

We conducted an informal user study to assess our prototype. Twelve volunteers, all male between the age of 20 and 30 were asked to perform some simple

modeling tasks within a virtual environment visualised in the PSD. All participants had limited experience with the PHANToM device. We first ensured that all our participants were pronouncedly right-handed, since in our setup the PHANToM device was placed at the right-hand side of the subjects [Hinkley et al., 1997b]. Possible participants had to score at least 16 out of 20 on the Edinburgh handedness inventory [Collins, 2003]. Three test persons were not sufficiently right-handed and were withheld from the experiment.

After reading the instructions, users were allowed to practice for about 5 minutes to get used to the environment and the task. The task consisted of firstly selecting one of three objects positioned at different places in the 3D world and then changing its texture (Figure 7.4(a)). An object could be selected by simply touching it; a new texture could be selected by choosing the correct item from the proprioceptive menu (see Figure 7.4(b)).



(a) Experimental Setup (b) Menu selection

Figure 7.4: Experimental task

Since we wish to compare the results with the experiment conducted in chapter 6, in a control condition all commands could also be given by using speech as described in chapter 6. Six participants were first asked to complete the task by just one modality (either voice or haptics) and next by the other. The conditions were evenly spread over these subjects using a counter-balanced repeated measures design. In a third trial, those users could repeat the task with a combination of the modalities they preferred. Since we are also interested in an evaluation of our solution without being compared with the speech condition, three participants where restricted to interact haptically, without being in touch with the speech modality.

During the test, all commands were manually logged. The dependent variables were ‘the chosen modality’ and ‘success of the command’. A command

was considered as unsuccessful when the user had to retry or had to look for an alternative. After the test, participants completed a short subjective satisfaction questionnaire. The questionnaire included items about the participant's preference for the speech or haptic menu, and the intuitiveness of the proprioceptive activation metaphor.

7.4.3 Results and Discussion

Proprioceptive Interaction Alone

When comparing the results of the participants that completed the task with direct manipulation, we see that about 5% of those actions went wrong, mostly because the user had chosen the wrong menu item by accidently sliding away while clicking. In the previous experiment, described in section 6, we conducted the same experiment, in which, with direct manipulation, the menu appeared somewhere in space around the user's 'look-at-position'. In this experiment, the average error rate with haptic interaction was about 13%. As can be seen from table 7.1, we can distinguish a decrease of the number of errors made, however, with a p-value of 0.10 we cannot proof the significance of this result. On the other hand, while in both tests a number of participants did not make any mistakes, the first test (with a floating menu) showed a much higher variance than the newly proposed interaction. When additionally performing an F test to compare variances, we found that the null-hypothesis with ratio=1 is rejected with a p-value=0.0075 (F=9.7). This means that the newly proposed solution reduces the variance in error rate.

From a subjective questionnaire, we can conclude that most participants find it intuitive to activate the menu by bringing the non-dominant hand to the PHANToM, which we can confirm from the observations, as well.

Proprioceptive Interaction Compared with Speech

To compare the newly proposed metaphor to our previous experiment, some of our participants had to perform the same test using speech commands. At the end, those users were asked to perform the same task using the modality they prefer. When we look at the error rate of the spoken commands, we observe an average of 28%, which is significantly higher than the error rate with direct manipulation (analysis with a student t-test with unequal variances has a result of 0.003 as shown in table 7.1). Surprisingly enough, from the

Table 7.1: Statistical results of the user experiment

(a) Comparison of the number of errors in the haptics modality with a menu floating in 3D and a proprioceptively activated menu

	Average	Variance
Floating Menu	13.3	171.7
Prop. Menu	4.3	17.8
df = 4	t = 1,50	p = 0.10

(b) Comparison of the number of errors between speech and haptic modality

	Average	Variance
Speech	28.4	197.8
Haptics	4.3	17.8
df = 6	t = 4,06	p = 0.003

subjective questionnaire, those people candidly prefer the speech modality. Although these results confirm our findings described in chapter 6, we have to question why users prefer a more error prone modality. Indeed, [Oviatt, 1999] disproved the myth that speech can be seen as a dominant modality. We believe the dominance of speech can be explained by the experimental behaviour of a user in a new environment, which can be confirmed by the following observation: when some participants forgot the exact command-phrase, they switched to the haptic modality as a back-up. After enabling the menu and homing to the correct menu item, just before clicking the menu item, they remembered the speech command. At that time, those users then left the direct manipulation modality and again used the spoken command.

Another important finding is the increasing number of errors of the haptic modality when using both speech and haptic modality. Indeed, when both modalities can be chosen freely, the error rate of the haptic interaction raises from 5% up to 10%. Studying our observations, we can learn that when participants have chosen to use speech as the main modality, they are not focused on the haptic pointer. When activating the menu, our metaphor places the menu right behind the virtual pointer, which, at that time, can be at an unexpected or inappropriate location. It is very likely that this can cause the higher number of errors and even the overall preference for the speech

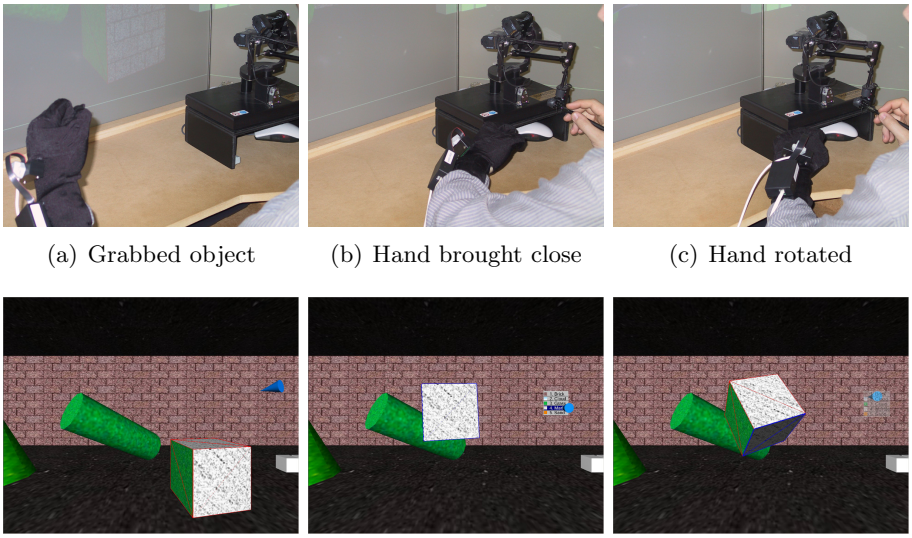
modality.

7.4.4 Summary of the Results

We can summarise that our proposed interaction paradigm, in a first proof of concept, has its benefits as it turns out to be a natural way of bringing a menu to the user. This assumption can also be motivated by the evaluation of one test person who told us that he liked the way the menu worked in the same manner as a PDA, although we did not mention this analogy in the instructions. On the other hand, as we want to compare the users behaviour to a known situation using speech input, we still see a strong preference for the speech modality which can be compared to the results of our prior experiments. Although, according to the conclusions we drew from the experiment described in chapter 6, we believe that a growing task-set and a more complicated application, as a result of the user's limited working memory, will change the preference to direct manipulation.

7.5 Object In Hand for Objects

In the previous section, we described the first implementation of the 'Object In Hand' metaphor, which could be used to activate a menu by using the non-dominant hand as a proprioceptive frame of reference. As an informal experiment showed the benefits of our approach, this section extends the 'Object In Hand' metaphor so that is is suitable for objects, as well. We believe that this could provide the user with a unified approach to activate menus and manipulate objects, and hence contributes to a more natural interaction with the virtual environment.



(a) Grabbed object (b) Hand brought close (c) Hand rotated

(d) Selected object (e) Object-in-Hand (f) Rotated object

Figure 7.5: Object-in-Hand metaphor

7.5.1 The Metaphor

Since it is difficult to manipulate 3D objects that are located freely in 3D space, we want a user to 'grab' a selected object and pull it out of its context in order to be manipulated. As the object is 'grabbed' by the non-dominant hand, it must then be possible to clutch and declutch and freely move and rotate the object in each direction using that hand. If the non-dominant hand is moved away, the manipulation is finished and the object should go back

to its original position. We believe this interaction metaphor is very suitable in situations in which the user wishes to manipulate an object outside of its context (such as colouring, texture painting or even simple object inspection).

A required condition, however, to make this manipulation metaphor work for objects, is that the user must be able to easily indicate what object to grab. Object selection therefore seems to be an obvious solution. Although it seems that, with this solution, we shift the 'object accessing problem' to a 'problem of selecting objects', currently, very suitable selection metaphors do exist (section 3.4.3). In section 7.6, we will investigate which selection is most suitable within the scope of the 'Object In Hand' metaphor.

Comparable to the solution presented in [De Boeck et al., 2004a], after an object has been selected, the user can 'grab' the object by bringing the fist of the non-dominant hand close to the pointing device held in the dominant hand. At that instance the selected object moves to the centre of the screen, where it can be manipulated by the dominant hand. In this example we allow the user to select the different faces from the object and choose a texture from the menu, positioned next to the floating object. Figure 7.5 shows the movements of the hand and the corresponding movements of the object.

In the next paragraphs, we will elaborate on the specific details of this two-handed interaction. To clarify those details, we use the NiMMiT notation, described in chapter 4. First we elaborate on the non-dominant hand grabbing and holding the object, next we shortly focus on the dominant hand manipulating the object, and finally we show how both hands work together in a synchronised manner.

Non-dominant Hand Interaction

To start, let us assume that the object can be selected by a suitable selection metaphor. As NiMMiT supports a hierarchical build-up, we will define 'object selection' as an hierarchical task (currently a black box) on which we will elaborate in section 7.6, later in this chapter. When the diagram in figure 7.6 is started, the state 'start' is activated, and immediately, as the system is 'idle', the selection hierarchical task is executed. As long as this task is not finished, the execution of diagram 7.6 is suspended. At the end of the selection, we store the result (the selected object) in the label 'selected'. Next, a state transition to 'Prepare OiH' (Object-in-Hand) is performed.

This state waits for a gesture recognition. The gesture is defined as a 'closed non-dominant hand brought in the proximity of the dominant hand'. When

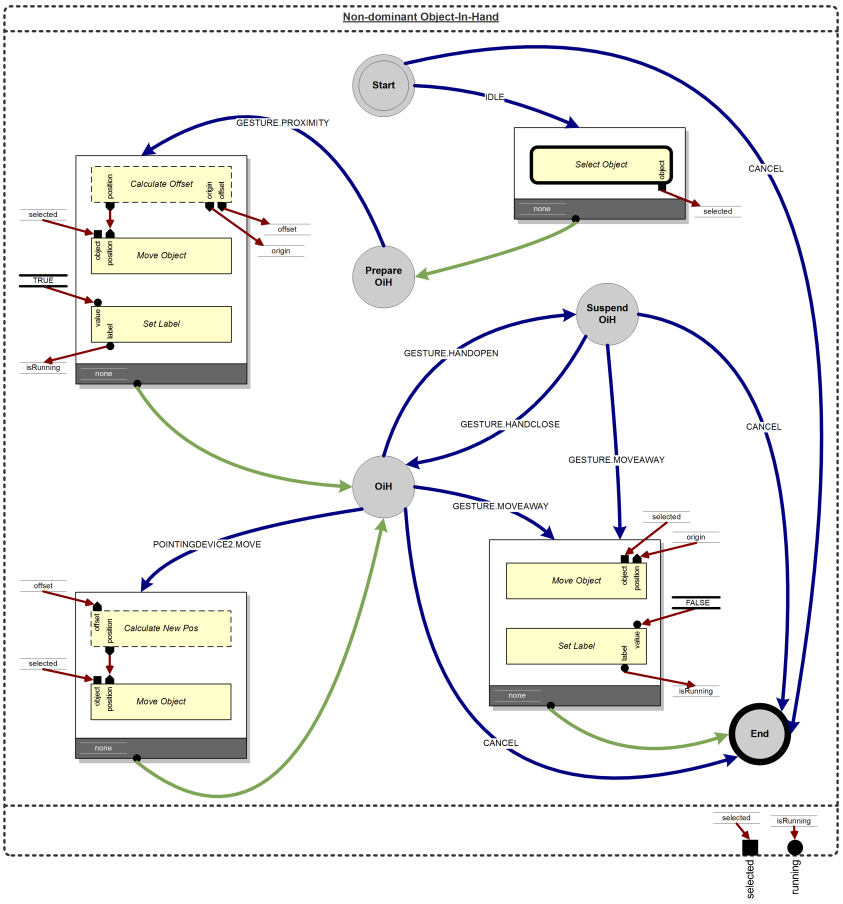


Figure 7.6: The interaction of the non-dominant hand

this event takes place, the second task chain is fired. The first task in this chain is a scripting task, which calculates a custom offset between the virtual position of the non-dominant hand and the centre of the world. The task requires the selected object as an input. The output is the original position of the selected object, the offset and the new position of the object. All these outputs are stored in labels. The next task moves the selected object to the given position, and the last task simply sets a label to true. This label is needed for the synchronization, which will be explained later in this section.

Eventually, the system ends up in the state 'OiH' and looks for one of the awaited events. When the user opens his/her hand (recognised as a gesture), a transition (without task chain) to the state 'suspend OiH' takes place. If

the hand is closed, the 'OiH'-state is activated again. These state transitions implement the functionality for clutching and declutching the object. 'OiH' also responds to movements of the non-dominant hand. In the corresponding task chain, a new position is calculated according to the movements of the hand, and the object is moved. Finally, both states respond to the gesture 'withdrawing the non-dominant hand'. In both cases, the activated task chain restores the object's original position and resets the 'isRunning' label to false.

Dominant Hand Interaction

As soon as an object is brought in position, the user can start manipulating it. In this example the user can select faces of the object by just touching it. Next, by choosing from a menu, the texture of the selected face can be altered.

At start-up, the interaction technique receives the selected object and a boolean as input (figure 7.7), and starts at the state 'manipulation'. This state responds to two events: a movement of the pointer and a 'texture change' event, coming from the menu. When the virtual pointer is moved, a task checks for collisions with one of the faces of the selected object. If a collision is detected, the correct face is stored in the label 'face'. In addition, the face is passed on to the next task, which selects it. Afterwards, a state transition returns the system to the initial state. The second task chain is responsible for handling the menu events. This chain contains a single task, which changes the texture. If the task receives no viable input at the optional 'face' port, the texture of the entire object is changed. Otherwise, only the specified face alters texture.

Merging Both Hands

The previous paragraphs elaborated on the interaction with the non-dominant and the dominant hand respectively. However, both hands do not work independently. The interaction of the dominant hand is active only while the interaction of the non-dominant hand is running. Therefore, both diagrams need a synchronisation mechanism.

As soon as the interaction of the non-dominant hand becomes active, it defines the label 'isRunning', which is directly routed to the output. The second interaction technique receives this label via its input (passed-on via a higher application level). Next, the value of the label is checked in the precondition of the technique. If the value is false, the precondition is not fulfilled, and the execution of the diagram is aborted. Otherwise, the interaction technique

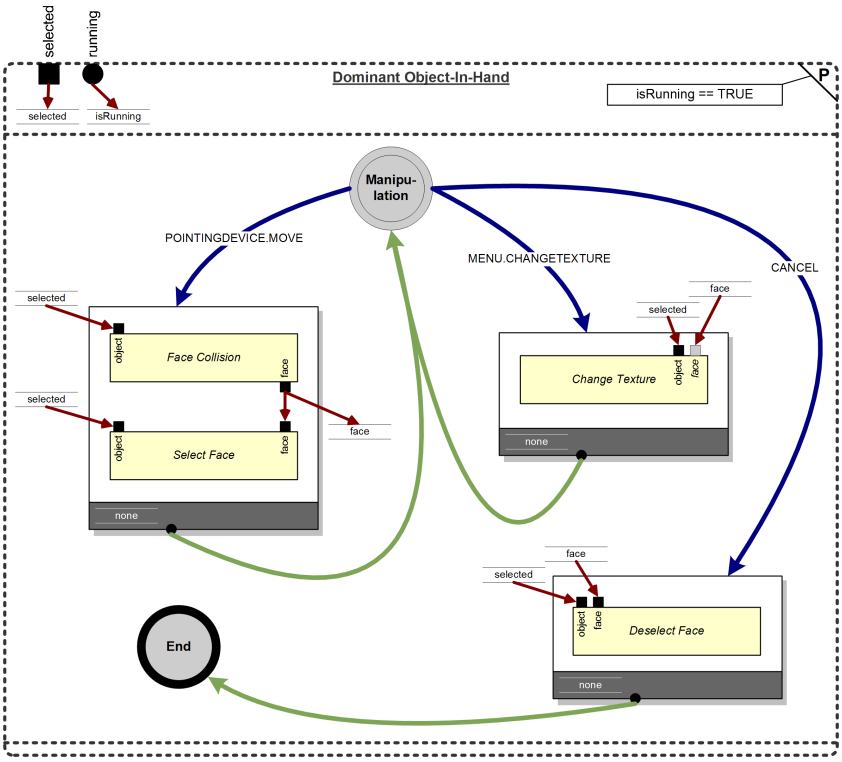


Figure 7.7: The interaction of the dominant hand

is started. Whenever the non-dominant stops its interaction, the boolean value is reset to false. Subsequently, the precondition of the dominant hand's interaction technique fails and the execution of the diagram aborts. Obviously, for more advanced synchronisation, the boolean value can also be used within the diagram, e.g. to control a conditional state transition.

7.5.2 Experimental Validation

Application Features

As the previous experiments were only based on a very limited set of possible tasks, in this experiment, we have chosen to evaluate the proposed metaphor within a more elaborated application. The application is a modeller which allows to create, move delete and change objects in a generated 3D world, visualised in the PSD. Following features are available:

- **Proprioceptive Menu:** The menu, activated as proposed in section 7.4 is used to choose the main commands within the application: creating, deleting, moving and rotating objects, activating selection mode, navigating, etc. The menu is activated by bringing a flat non-dominant hand close to the dominant hand (in contrast to the fist in order to grab an object).

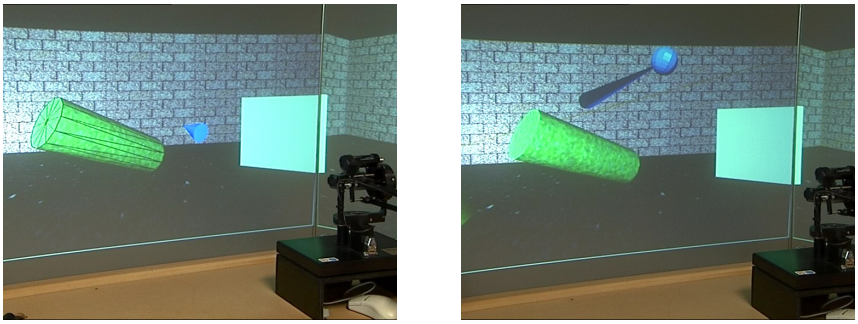
- **Navigation:** To navigate within the environment, we integrated the 'Camera In Hand' metaphor, as covered in chapter 5.

- **'Small Scene Manipulation':** This metaphor, also described in [De Boeck et al., 2004a], can be seen as some kind of automatic teleportation. When the target is relatively far from the user's viewpoint, interaction techniques such as AAAD or Go-Go can be applied for manipulation. Still, in case of objects that are far away, with these solutions, the visual feedback is a bottleneck. Another approach is to manually navigate to the target and 'zoom in' to the place of interest. Accurate navigation in a 3D world, however, is not always obvious, and often, after manipulating the object, the viewpoint must be restored to the original position, implying a double navigation task. Opposite to our solution for object-manipulation, in which the object is brought to the user, the evaluation application also proposes the 'Small Scene Manipulation' which is a technique to easily move the user to the selected object, and back to the original position. To minimise the risk of being lost in a teleportation, the camera smoothly moves to the new position.

- **Selection:** Selection is a essential part of the 'Object In Hand' metaphor, however, later in this section we will search for the most suitable solution. In this application, we provide the user with two well known selection metaphors: virtual hand and cone selection, as is illustrated in figure 7.8. Selection mode is accessed via the proprioceptive menu.

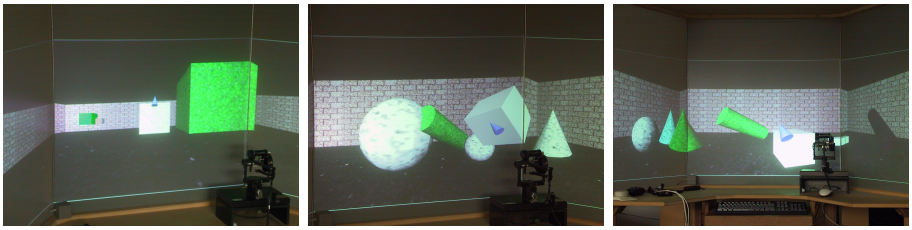
Experimental Setup

To assess the proposed interaction technique within the scope of the modeling tool, we have conducted a usability test. Eleven student volunteers were asked to participate. Nine of them were right handed while two of them were left-handed. Although our setup is right-handed, this time we allowed some left-handed subjects to participate, only since we also were interested on how



(a) Virtual Hand selection (b) Cone selection

Figure 7.8: Object selection



(a) Scene 1 (b) Scene 2 (c) Scene 3

Figure 7.9: Scenes for the three tasks of the experiment

left-handed people would evaluate the application. All subjects had extensive experience with computers, but none of them had any experience in using the PHANToM or multimodal virtual environments. A demo video of about 10 minutes that explained the possibilities of the application was presented to all subjects. After that, using a checklist, they got some time to try all the functions themselves in practice. From the start, all users were encouraged to think aloud, while everything was recorded on video. After a test-period of an average of 10 minutes, subjects had to perform three tasks (figure 7.9). Afterwards, users were asked in a questionnaire for their personal appreciation of the interaction technique and the application in general.

The first task is less important in the scope of this work, but for the sake of completeness, we would not omit it in this thesis. The task evaluated the 'Small Scene Manipulation'. A scene with two groups of two cubes was presented (figure 7.9(a)). Users had to put the small cubes on top of the large cubes. The users were not restricted in the way they wanted to solve the

problem. Nor were there any requirements of accuracy: the task was finished when users decided to. The front-most cubes could be accessed directly from the current camera positions. The cubes on the backside of the scene were out of reach so that our test persons had to find a solution themselves: either they could use ray selection and 'Small Scene Manipulation', or they could use normal navigation and normal selection or any combination of them.

The second task was designed to evaluate the 'Object in Hand' manipulation. Users were asked to colour four faces of a cube with a different texture. Therefore, they had to select the object and 'grab' it with their left hand (figure 7.9(b)). Since we found it obvious that this interaction method is faster than manually navigating around an object, constantly switching between modes, no alternative for the manipulation was presented. We observed the subjects using this metaphor, so that we could evaluate the strengths and the weaknesses of the interaction.

In the third task, we evaluated the integration of all metaphors within the application itself. Subjects were presented with a scene containing several objects divided over all screens of the PSD (figure 7.9(c)). Several tasks had to be fulfilled such as adding objects, moving objects and colouring faces. Some tasks could be executed directly, other tasks required navigation, 'Object in Hand' manipulation or 'Small Scene Manipulation'.

7.5.3 Results and Discussion

If we first consider the basic interaction with the menu, we could observe that several subjects initially had trouble coordinating the appearance of the menu. Most of the time, users initially forgot to bring the virtual pointer to an appropriate position, since the menu will appear at that location. Six out of eleven subjects evaluated the menu-interaction positively. The other subjects criticised the fact that they first had to bring the virtual pointer into position. We suppose that the sensitivity of the PHANToM device within the PSD can be one of the causes of these remarks. Indeed, to allow the user to reach the entire proximity of the visible world on all three projection-screens, a very high sensitivity for the PHANToM is required. This has a side effect that a very small (involuntary) movement of the PHANToM has a large effect on the virtual pointer and even moves it behind the camera position.

Next, the 'Object in Hand' metaphor was evaluated as beneficial. Users report that this metaphor is a very natural and intuitive way to interact with the object. From our subjective questionnaire (figure 7.10), we see that eight

users found it intuitive to grab an object. Eight users found it easy to rotate the object to see other faces. Finally, also eight people were positive about the interaction between the object and the texture menu next to the object.

As we were observing the use of the 'Object in Hand' technique, we could notice that some users initially had trouble in coordinating the movement to bring a closed hand to the PHANToM in contrast to the flat hand necessary to activate the menu. In addition, the coordination of clutching and declutching an object also caused some problems. During the practicing period of 10 minutes, those problems mostly disappeared and the technique was appreciated. Another remark we could note is that one test person complained about the position of the fixed menu when in 'Object in Hand' mode. It is true that the menu has been positioned on the right-hand side of the object, at the same z-value as the centre of the object. This makes that, when manipulating the front-most faces of the object, the depth of the menu must be sought for. However, placing the menu more to the front will introduce the same problem in the opposite direction.

There are some other observations that we could make in the margin of this usability test. First, the 'Small Scene Manipulation' was avoided by most users, even in spite of the task which was designed to use this metaphor. Possibly, the fact of feeling to loose control being moved to an unknown, or maybe unappropriate position can be a reason. On the other hand, we also observe that even the other alternative to complete the task hasn't been used either: at least five subjects even didn't try to navigate to the objects, even if they were too far to manipulate. Instead, they used ray selection and moved the object from the current position, of course degrading accuracy.

And surprisingly enough, we can also conclude that subjects try to avoid the ray selection as much as possible. When thinking aloud some subject even complained to have to switch to ray selection. We suspect this as a result of the combination of the user's grip of the PHANToM stylus, the position of the PHANToM device and the rotations to make with the wrist which results in uncomfortable and thus less controllable poses. Next, there seem to be no significant difference in behaviour or appreciation of the application between right-handed or left-handed subjects, although our left-handed subjects seemed to take more time for the practicing phase, as could be expected as a direct result from the larger amount of motoric errors related to the non-dominant hand. Their accuracy and behaviour when finally performing the test did not differ much. This finding however cannot be validated, since only two subjects were left-handed. Another remark is that only one person com-

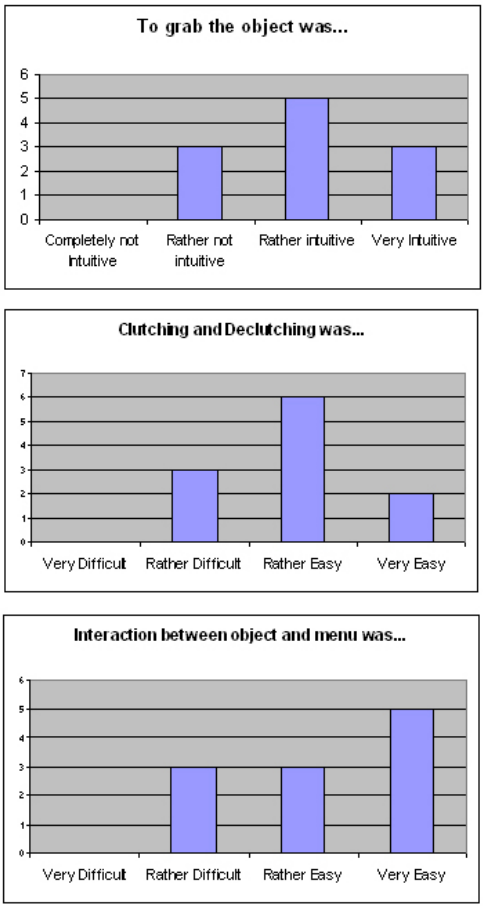


Figure 7.10: Results of the subjective questionnaire

plained about tiredness holding the PHANToM device, while users indeed had to sit down for half an hour while both hands were in action without being supported all the time. Finally, the setup in our PSD, with three projection screens certainly has its benefits to improve the feeling of being involved into the world. However, when users want to interact with objects, or with the menu, they seem to have a preference to interact on the middle screen and avoid interacting on the side screens.

In summary we can conclude that, the 'Object In Hand' metaphor is a promising solution in order to allow users to use their proprioceptive knowledge to activate control elements (like menus) or (temporarily) grab objects. However

selecting objects, and especially distant objects still causes troubles. In our setup, the ray selection is not really appreciated, but the alternatives such as navigating are not popular either. In the next section, we will search for a suitable solution to integrate object selection within the 'Object in Hand'.

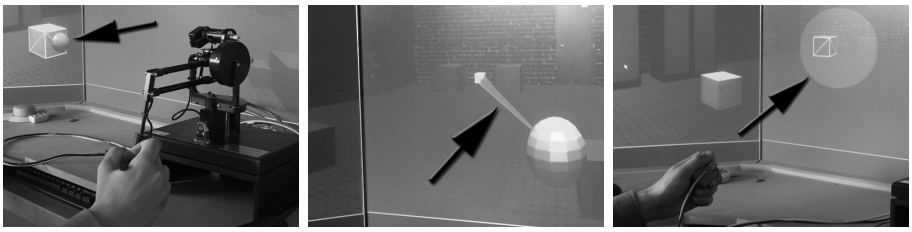
7.6 Selection Metaphors

7.6.1 Existing Selection Metaphors

As selection is one of the basic tasks in nearly every application, it is not surprising that a lot of work can be found about selection techniques. A comprehensive overview, with references to the original authors of the most common techniques, can be found in [Bowman et al., 2005], the most important solutions are summarised in section 3.4.3. In this section we will elaborate on three of them: virtual hand, ray or cone casting and aperture selection, as they appear at first sight to be most suitable to be integrated with the 'Object In Hand' metaphor. In the next paragraphs, we will shortly summarise how the metaphors work. In addition, we also explain the corresponding NiMMiT diagrams which have been designed for the implementation of the metaphor.

Virtual Hand

This interaction technique is by far the most widely known selection technique. A virtual representation of the user's hand or input device is shown in the 3D space (figure 7.11(a)). By moving a tracked hand or input device, the virtual representation is moved accordingly. When the virtual hand intersects an object, the object becomes selected. This metaphor has the advantage to be very simple and intuitive, since it is very similar to touching an object in real life. One of the drawbacks is that, dependent on the supported device, the technique can be tiring with repetition. The main drawback, however, is the limited workspace in which objects can be touched. As in real life, distant objects cannot be touched. Solutions such as 'Go-Go' [Poupyrev et al., 1996]



(a) Virtual Hand (b) Ray Casting (c) Aperture Selection

Figure 7.11: Screenshots of three selection metaphors.

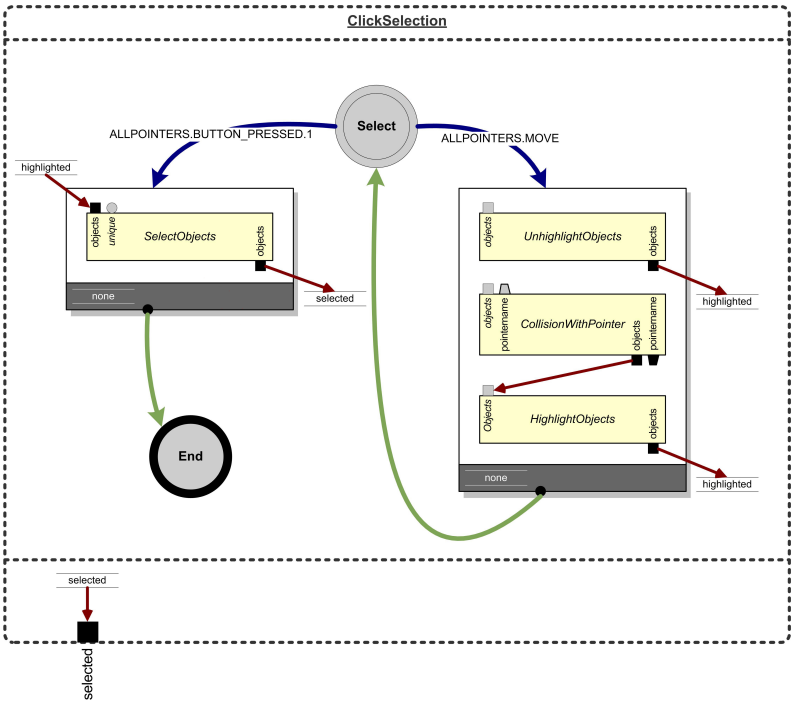


Figure 7.12: NiMMiT Diagram of Virtual Hand and Ray selection

try to solve this problem at the cost of less accuracy for distant manipulations. Alternatively, a navigation task has to precede the selection task, but dependent on the application, this is not always desirable either.

Figure 7.12 shows the NiMMiT diagram of the virtual hand selection. The interaction technique starts in the state 'Select', which reacts to following events: a pointer movement and a click. Each time the pointer moves (and the button is not pressed), the rightmost task chain is executed. This chain contains three consecutive, predefined tasks: unhighlight any highlighted objects, check for collisions and highlight an object if necessary. The first task has an optional input; if not provided, all highlighted objects are released and the output returns an empty list. The second task contains two optional input ports, providing the objects and pointers that should be taken into consideration by the collision detection. If the optional inputs are connectionless, default values are used: collision is calculated between all pointers and objects in the environment. The colliding objects, if any, are obtainable through the output port.

The final task, highlighting the colliding object, has an optional parameter, indicating the objects that have to be highlighted. If the previous task did not detect a collision, this task performs no operation. Otherwise, the objects are highlighted and the results are stored in the label 'highlighted'. Finally, a task transition returns the system to the state 'Select'.

While the system resides in the state 'Select', a click event causes the leftmost task chain to be executed. It contains only one task: selecting an object. If the previous chain was successfully completed, the task selects the highlighted object, received by connecting the label 'highlighted' to the input port. Additionally, the task stores the selected object in a new label, 'selected'. In case the label 'highlighted' contains no viable value (e.g., no object was highlighted), the chain is aborted, returning the system to the state 'Select'.

When the second task chain finishes successfully, a final state transition occurs. The system moves to the ending state and the selected object is outputted, using the label 'selected'. At that moment, the interaction technique is completed.

Ray Casting or Cone Casting

This interaction metaphor mimics the manipulation of a flash-light or laser pointer, in order to allow the user to select distant objects [Liang and Green, 1994]. From the virtual pointer, a ray or a cone is casted into the world (figure 7.11(b)). The closest object that intersects with the ray or cone becomes selected. In this manner, distant objects that cannot be touched with a virtual hand can be accessed. However, difficulties arise when selecting far and small objects. In several applications, ray casting turns out to be a good solution, but compared to a virtual hand, we could conclude from our former experiments that users rather try to avoid it [De Boeck et al., 2004a]. A formal comparisons between virtual hand and ray selection (using the dominant hand), has been described by Poupyrev et al. [Pouprey et al., 1998]. Here, the ray casting shows a slightly better performance.

Concerning the implementation of the Cone Casting metaphor, the NiMMiT diagram is equal to the virtual hand. The only difference is the shape of the virtual pointer, and hence the result of the collision detection step. With the virtual hand, the cursor is a simple sphere; with Cone Casting, we replaced it by a sphere with a cone attached to it.

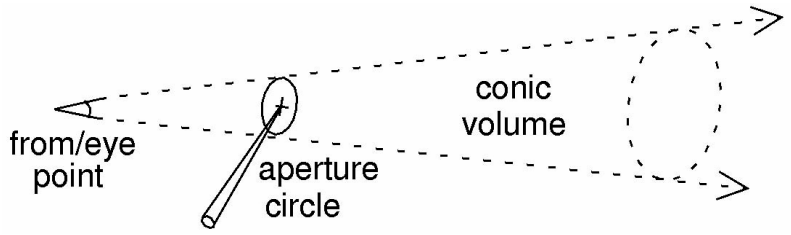


Figure 7.13: Schematic overview of the Aperture Selection (from [Forsberg et al., 1996])

Aperture Based Selection

The aperture based selection [Forsberg et al., 1996] defines a cone with its apex at the user's eye point. The cone runs through the aperture, a circle floating in the world, parallel to the projection plane. By moving the aperture according to the X or Y axis, the cone is changed accordingly. By moving along the Z axis, the width of the cone is adjusted, as the aperture always keeps the same size. This is schematically shown in figure 7.13. The object closest to the user, intersecting the cone, becomes selected. From the user's viewpoint, an object becomes selected when its projection falls within the aperture (figure 7.11(c)). In our opinion, the aperture selection keeps the advantages of the ray or cone selection metaphor, since it is based on the same technique of directing a cone into the world. However, the cone is not controlled by rotations, but by a translation instead, which makes it more controllable. On the other hand, still difficulties exist when accessing far and small objects.

Figure 7.14 shows a NiMMiT diagram of the Aperture Selection. The diagram begins in the state 'Start', from which immediately, an idle-event executes the first task chain, which initialises the aperture cursor. Next, the diagram arrives in the state 'Select' in which a 'pointer move' or a 'button press' event is awaited. When the pointer moves, the leftmost task chain is performed. In a first task any highlighted objects are unhighlighted. Next, a 'custom task' changes the aperture on the screen according to the cursor movements, as the visualisation of an 'aperture' is not supported in the framework. This task returns the cone running through the aperture and passes it to the next task, calculating the collision between the cone and any other object in the scene. The result of the collision detection, a list of objects, is given to the next task. This scripting task calculates which object from the list is closest to the current viewpoint. Finally, the chosen object is sent to the 'highlight' task

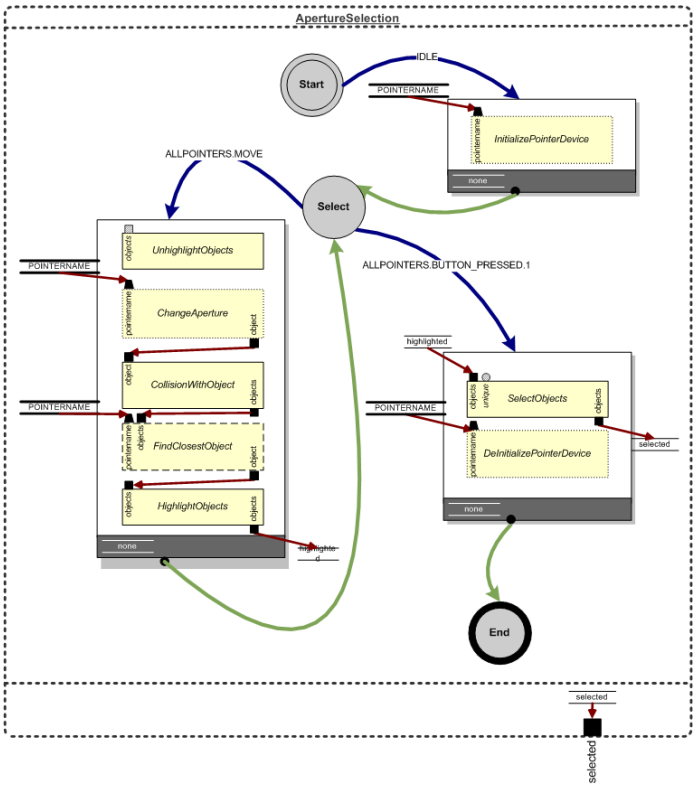


Figure 7.14: NiMMiT Diagramma of Aperture selection

and the result is stored in the label ‘highlighted’. After this task chain has been completed, a state transition, to the state ‘Select’ is executed. As soon as a button is pressed, the rightmost task chain is activated. Here the object, stored in the label ‘highlighted’ is selected, and the selected object is stored in the label ‘selected’. If this label does not contain a valid object, because the former task chain did not completed successfully, the current task chain is aborted and the state ‘Select’ is restored. Finally, the last task in this task chain does some deinitialisation, before the IT moves to the end state. The IT sends the selected object to its output via the selected label.

7.6.2 Experimental Approach

Motivation

As depicted in section 7.5, the 'Object in Hand' metaphor provides the user with a very intuitive way to 'grab' an object in order to manipulate it. However, before grabbing, the object of interest must be selected first, which shifts the 'object accessing problem' to a 'selection problem'. In our former application, we provided the user with both a virtual hand and a cone selection metaphor, but none of them turned out to be ideal because of the aforementioned reasons: the virtual hand suffered from a limited workspace and cone casting, mainly controlled by rotations of the wrist, while sitting in front of a PHANToM device, turns out to be difficult for some users.

In this section, we search for a better alternative in order to solve the selection problem as part of the 'Object in Hand' metaphor. As 'Object in Hand' is a two-handed interaction technique, we believe the non-dominant hand can play an important role in this solution.

Hinckley [Hinkley et al., 1997b] found that for complex tasks, specialisation of the roles of the hands is strong and important, but for easy tasks, reversing the roles of the hands would not have a large effect. Because at first sight, selection turns out to be a precise task, it is generally accepted to be performed by the user's dominant hand. However, in real life, since the non-dominant hand is used to hold an object, it is very likely that this hand is also used to pick the object out of its context. As we here suggest the similarity between 'picking out an object in the real world' and 'selecting an object in the virtual world', we came up with the idea to involve the non-dominant hand in the selection task. As the every-day grab and hold operation is very similar to the 'Object in Hand' metaphor, we believe it is worth measuring the performance of the aforementioned selection techniques using the non-dominant hand.

We conducted a formal experiment, which is described in the next section. Here we compare the selection metaphors and the performance of the dominant and the non-dominant hand. As the goal of the experiment is to use the results directly as an improvement of the 'Object in Hand' metaphor, we have chosen the input devices so that the results immediately fit our setup, although we are aware of the fact that this has its influence on the generality of this experiment.

Since force feedback improves the experience with the virtual hand selection, the PHANToM device is used for this metaphor. To compare with our former results, the PHANToM is also used for ray selection. For the aperture

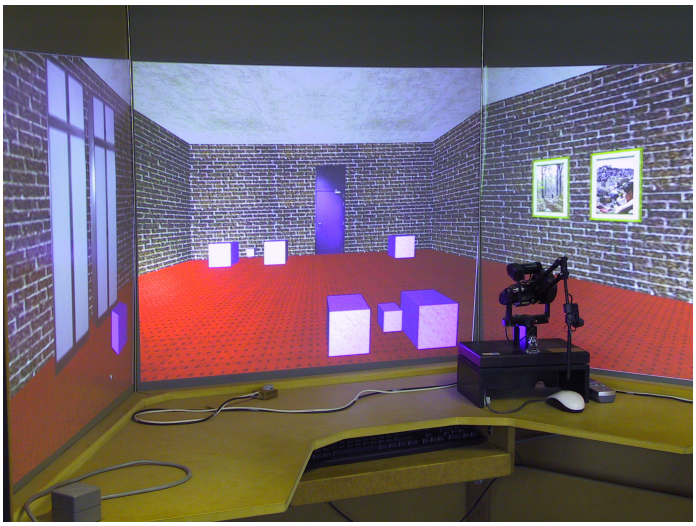


Figure 7.15: View of the experimental scene in the PSD

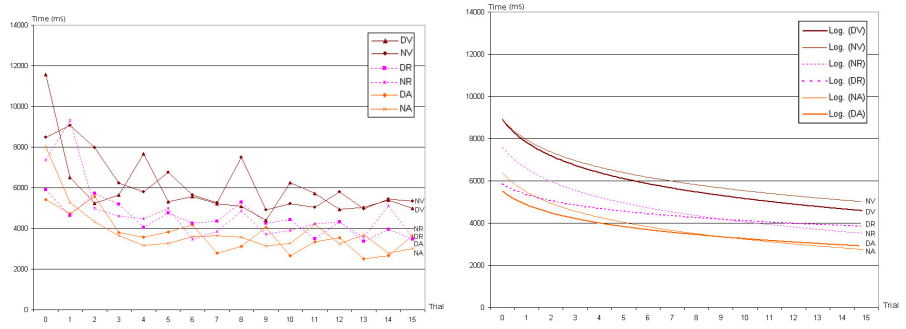
selection, we have chosen to use a tracker instead of the PHANToM, since this would benefit the integration, especially when we want to use it with the non-dominant hand.

Experiment

In the conducted experiment, also described in [De Boeck et al., 2006a], the three selection metaphors, virtual hand, cone selection and aperture selection, were tested using the dominant and the non-dominant hand. The six conditions were counterbalanced using a Latin Square design.

Twelve volunteers, ten males and two females, with little or no experience in 3D interaction, were asked to select a series of small and large objects. Some objects were positioned close by, others further away, as shown in figure 7.15. The small objects were one third of the size of the large objects. The distant objects were placed at the far back side of the workspace of the virtual hand, while the objects positioned close by were positioned near the front side.

All our subjects were between the age of 22 and 31; only two of them were left handed. From a predefined list, certain objects in the scene were highlighted in an alternating way (far-close, small-large) in such a way that no two similar objects would be highlighted subsequently. The users were asked to select



(a) Raw data. (b) Logarithmic trend line.

Figure 7.16: Average results for all subjects per trial.

the highlighted object as efficient as possible using the offered metaphor and the demanded hand. For each condition, the subjects had to perform 16 trials from which the first 4 trials were considered as a practice. After a selection had been carried out, audio feedback was given to indicate the result (success or not). During the test, the time and result of the selection were logged. We also logged whether the object was small or large, and wether it was positioned far or close by. After the test, each volunteer was asked to give his/her subjective impressions.

7.6.3 Results

General results

Figure 7.16(a) shows the average results of the completion times per trial over all users for each condition. We use the following abbreviations for each condition:

- **DV**: Dominant Hand, Virtual Hand
- **DR**: Dominant Hand, Ray Selection
- **DA**: Dominant Hand, Aperture Selection
- **NV**: Non-Dominant Hand, Virtual Hand
- **NR**: Non-Dominant Hand, Ray Selection
- **NA**: Non-Dominant Hand, Aperture Selection.

	Time (ms)		Time (ms)	P-value
DV	5328.68	DR	4207.77	<0.001
DR	4207.77	DA	3203.95	<0.001
NV	5678.80	NR	4228.80	<0.001
NR	4228.80	NA	3370.39	<0.001
DV	5328.68	NV	5678.80	0.38024
DR	4207.77	NR	4228.80	0.94159
DA	3203.95	NA	3370.39	0.28560
DV	5328.68	NA	3370.39	<0.001
DR	4207.77	NA	3370.39	0.00140

Table 7.2: Average completion times per condition.

To clarify the results for a first observation, a logarithmic trend line is calculated in figure 7.16(b). Here we see that the virtual hand metaphor is slower than the ray casting, which at its turn is slower than aperture based selection. It also appears that there is little difference between the performance of the dominant and the non-dominant hand.

Table 7.2 compares the different conditions using one way ANOVA. While calculating the averages, the first four trials of each user were left aside, as these were meant for practicing. Considering the dominant hand, ray selection turns out to be significantly faster than virtual hand selection. Aperture based selection at its turn is significantly faster than ray selection. The same is true for the non-dominant hand. When comparing the performance of the dominant and the non-dominant hand, it is not a surprise that the non-dominant hand is slightly slower than the dominant hand, but this result is far from significant with p-values well above 0.20. If we look at the results which are directly applicable to our setup, we see that the aperture based selection using the non-dominant hand is even significantly better than both virtual hand and ray selection with the dominant hand. The p-values are respectively 7E-13 and 0.001.

Errors

In table 7.3, the absolute and relative number of errors per condition are depicted. We can see that, using DR and NV, the number of errors is at its highest. The NA-condition appears to generate the lowest number of errors. With a chi-square test value of 0.41, however, the differences are not significant.

Even if we compare the best condition (NA) with the worst (DR), a chi-square value of 0.063 is not significant. Hence we can conclude that all the conditions perform equally well regarding the number of errors.

		Errors	Samples	%	Time (ms)
D	V	7	132	5.30%	5328
D	R	14	132	10.61%	4207
D	A	9	132	6.82%	3203
N	V	12	132	9.09%	5678
N	R	10	132	7.58%	4228
N	A	6	132	4.55%	3370

Table 7.3: Number of errors per condition.

Small vs Large Objects, Far vs Close Objects

Looking at the behaviour of the metaphors with respect to the size or position of the object, other interesting conclusions can be drawn. Here, we notice no significant difference between the dominant and the non-dominant hand either. Therefore, we put the measurements of both hands together. Table 7.4 shows that all metaphors are significantly faster selecting large objects.

If we look at the distance of the object (table 7.5), surprisingly, there is no difference using the virtual hand metaphor. Ray selection, however, seems to be significantly slower when selecting objects which are close to the user. We believe this is due to the fact that the rotation of the ray plays a more important role when the objects are close by. In our former work, we already discovered that users try to avoid the rotations with this metaphor. Finally, the aperture based selection appears to be faster for the close objects, although this difference is not significant.

	Small	Large	P-Value
Virtual Hand	5973.09	4970.91	0.012
Ray Selection	4465.84	3767.72	0.011
Aperture Selection	3791.92	2785.42	1E-11

Table 7.4: Comparison between small and large objects

	Far	Close By	P-value
Virtual Hand	5532.97	5495.43	0.925
Ray Selection	3791.88	4645.68	0.003
Aperture Selection	3410.02	3164.33	0.114

Table 7.5: Comparison between far and close by objects

Subjective Results

After the user completed all assignments, they were asked to complete a small survey, asking for their subjective perception of the different metaphors. Subjects had to rate their 'amount of agreement' with the given statement ('It was easy for me to select objects using the following metaphor') on a scale from 0 to 10, with 0 indicating a total disagreement. This is shown in table 7.6. We see a higher agreement, both in the dominant as the non-dominant hand condition, for the aperture selection. If we statistically compare DV with DA and NV and NA, we see a significance in both cases (respectively p=0.04 and p=0.03), which allows us to conclude that our subjects found it easier to select objects using aperture based selection.

Condition	Score
DV	6.50
DR	6.33
DA	**8,08**
NV	4.25
NA	4.08
NA	**6,50**

Table 7.6: Response to the question: 'It was easy for me to select objects using the following metaphor'.

Secondly, we asked for the users' preference when they could choose one of the metaphors for their dominant and their non-dominant hand. As can be seen from figure 7.17, for the dominant hand, five subjects preferred the virtual hand, three chose ray selection and four the aperture selection. This result is non-significant compared to the expected values (chi-square=0.77). For the non-dominant hand, no one preferred ray selection, three subjects chose virtual hand, while nine preferred aperture selection. With a chi-square value of 0.005, this choice is significant.

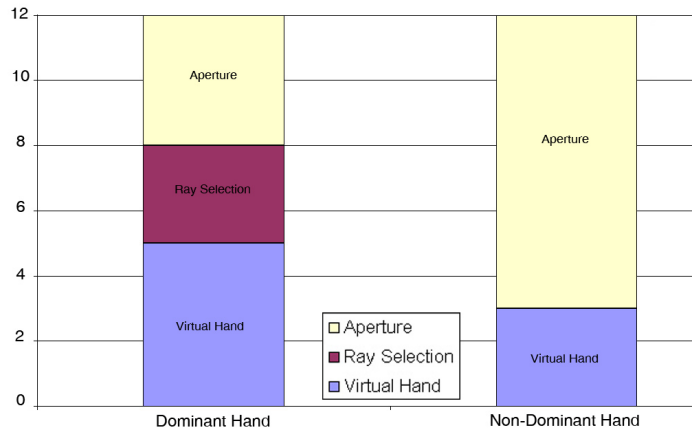


Figure 7.17: Subjective choice for the dominant and the non-dominant hand.

Summary of the results

The aperture selection is significantly faster compared to ray selection, which at its turn is significantly faster than virtual hand selection. This is both true for the dominant hand, and for the non-dominant hand. When comparing the performance of the metaphors, we find no significant difference between the two hands. Surprisingly enough, aperture selection in the *non-dominant hand* is even faster than the virtual hand or the ray selection using the *dominant hand*.

When looking into the number of erroneous selections, we cannot find a significant difference, allowing us to conclude that all the metaphors independent from the hand behave equally regarding the number of errors.

It turns out that all metaphors are significantly better for selecting large objects, but the results are less unambiguous when looking at the object's position. While there appears to be no significant difference between the selection of far and close objects using the virtual hand or the aperture selection, the ray selection turns out to be worse for close objects.

Finally, subjectively spoken, users claim to make selections easier using the aperture based selection. This is true for the dominant and the non-dominant hand. When users are asked to make a final choice, there is no pronounced choice for the dominant hand, but aperture selection is preferred for the non-dominant hand.

From these results, we can conclude that, in our experiment, aperture selection turns out to be better than the other metaphors. We believe this is true because this metaphor combines the benefits of the others, while eliminating the drawbacks. Moreover, from the user's point of view, the interaction basically appears to be 2D, eliminating the 3D overhead. This result allows us in the next section to integrate the aperture selection, using the non-dominant hand within the 'Object in Hand' metaphor.

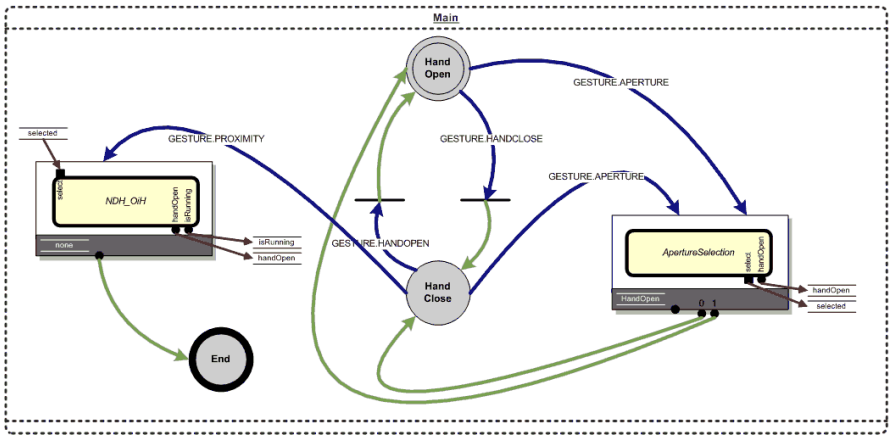


Figure 7.18: NiMMiT diagram of the OiH metaphor with selection

7.7 Object In Hand with Selection

7.7.1 Description

In this section, we have combined the 'Object In Hand' metaphor as described in section 7.5, together with an Aperture Selection controlled by the non-dominant hand. For the non-dominant hand, two gestures already were recognised: either the user could bring a flat hand to the dominant hand, in order to activate a menu. Or the user could bring a fist close to the dominant hand, simulating an object 'grab', bringing an object in a central position of the screen.

In the integration, additional to the existing gestures, the user can hold the thumb and the index to each other (aperture-posture), activating the aperture selection. By moving the hand freely, the aperture on the screen is moved accordingly. If the user moves the hand in the Z-direction, the aperture is brought closer or further away, giving a visual impression of changing the size of the aperture circle. Objects which are in the aperture circle become highlighted. As soon as the user wants to confirm the selection, the non-dominant hand can be closed to a fist. Now, the user can move his fist to the dominant hand continuing with the remainder of the 'Object in Hand' metaphor. The NiMMiT diagram of this interaction is shown in figure 7.18; for clarity the activation of the menu by bringing a flat hand close to the dominant hand is omitted.

7.7.2 Evaluation

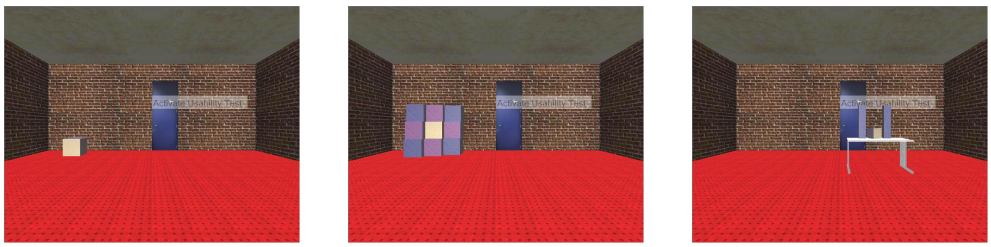
Considerations

Although we have shown that the 'Object in Hand'-approach has its benefits and that the aperture selection with the non-dominant hand is faster than the common solutions (such as virtual hand or ray selection) with the dominant hand, it is not obvious to formally evaluate the proposed solution. Indeed, we can assume that taking together a selection and a manipulation metaphor is faster in any way, compared to the separate solutions.

In order to evaluate our solution, we have chosen to measure how it behaves with respect to the scene complexity. We consider the outcome of the experiment as beneficial if the metaphor does not degrades significantly as the complexity raises. Additionally, the comments of the users from a small survey, together with our observations during the test, can be of a value for further improvements of the solution.

Experiment

Twelve volunteers, all between the age of 23 and 45, eight males and four females, were asked to participate in the test. All subjects were right handed so that they could fit our right-handed setup. After reading the instructions, each user was asked to select a coloured object and change the texture of a given face. This had to be done in 6 different scenes, increasing in complexity. The first three scenes were for practicing purposes, the last three (shown in figure 7.19) were taken into account for the test. As can be seen from the picture, the first scene only contains one box, the second scene contains a box



(a) Scene 4 (b) Scene 5 (c) Scene 6

Figure 7.19: Screenshots of three scenes in the test.

Table 7.7: Average time per scene

	Scene	Time (ms)	Stdev
Selection	Scene 4	3764	1511
	Scene 5	4631	1348
	Scene 6	4437	983
	Total	4304	1287
Manip	Scene 4	17620	11113
	Scene 5	14808	7635
	Scene 6	17074	8358
	Total	16484	8996

in the middle of other boxes. In the last scene, the box is halved in size, put on a table and accompanied by a cylinder on each side. For each scene, the time spent to select the object and the time necessary to change the texture were logged, together with correctness of the actions' outcome. For the logging, we applied NiMMiT filters and probes as described in 4.6 and [Coninx et al., 2006]. After the users completed the test, they were asked to fill out a little inquiry, to poll for their subjective impressions.

To recognise the gestures of the non-dominant hand, subjects wore a modified pinch-glove equipped with a magnetic tracker. The user's dominant hand operates a PHANToM force feedback device. The postures of the hand are recognized by the contactors of the pinch-glove: by closing the thumb and the index, contact is made and the aperture gesture is recognized. By closing the entire hand, another contact is made to recognize the fist posture.

After selecting the object using the aperture selection technique, and bringing it in a central position, the texture of the frontmost face had to be changed by selecting the face and choosing a texture from the menu. The manipulation of the object is carried out with the PHANToM device, while 'holding' the object with the left hand.

7.7.3 Results and Discussion

Table 7.7 shows the average time over all correct trials of all users, for both the selection and the manipulation. Table 7.8 shows the number of correct actions with respect to the number of errors.

Although we expected scene 6 to be the most difficult for the selection, the objective results show that scene 5 causes more troubles. However, when we

Table 7.8: Number of errors per scene.

	Scene	#Correct	#Error
Selection	Scene 4	14	3
	Scene 5	15	10
	Scene 6	20	5
	Total	49	18
Manip	Scene 4	12	2
	Scene 5	12	3
	Scene 6	11	4
	Total	35	9

compare the values statistically using ANOVA, there is no significance: even between scene 4 and scene 5 (the lowest and the highest value), we receive a p-value of 0,11. If we look at the number of errors of the selection, it is true that scene 5 performs worse than the other, but the chi-square value between scene 4 and 5 is only 0,08, while the overall chi-square of the entire matrix is 0,17.

The same is true for the manipulation part of the interaction. We expected scene 6 to perform worst, because of the smaller object, but when applying ANOVA, even between scene 5 and scene 6, the p-value is as high as 0,50. Moreover, a chi-square test to find significance in the number of errors provides us with values between 0,40 and 0,70.

After the subjects completed their test, they were asked to fill out a subjective questionnaire. From the results, we could learn that most subjects were neutral or slightly positive about the interaction technique. Not surprising, is that most subjects agree that the interaction is rather tiresome in terms of holding the hand unsupported in the air in order to select or hold and manipulate the object. However, in section 7.5, we found that the the 'Object in Hand' without selection, was not too tiresome for most users.

From our observations, we could conclude that our modified pinch-glove worked fine, although some people had difficulties not to release the contacts (pinches) when closing the hand after the selection. This should be solved by a better tracking of the hand's postures. We did some experiments with a 5td-glove, but dependent on the size of the user's hands (in spite of calibration), there was too much variance between users. Finally, we also observed that when closing the hand, users inadvertently move there hand to the right. This explains why scene 5 performs worse for the selection, as the objects are stacked

together. This can be solved by some user assistance, which remembers the previous position, or holds the aperture at the current place when closing the hand.

Summary of the results

In summary, we can conclude that the complexity of the scene has no significant influence on the performance of the selection, although we can keep in mind that the number of errors raised in scene 5. Neither has a smaller object an influence on the performance of this particular manipulation task.

Although our solution looks promising, we also could learn that fatigue caused by holding the arms in the air is a drawback. To improve this, we will have to look for a solution to support the user's arms, although this is not obvious, since the PHANToM force feedback in the PSD limits the possible positions. Other improvements may be required for a better recognition of the gestures (as independent as possible from the size of the user's hand) and to prevent inadvertently moving the hand when making a fist.

7.8 Summary

In this chapter, we investigated how two-handed interaction, making optimal use of the user's proprioceptive knowledge, could be achieved in our setup. In a first attempt, we aimed for a solution with two PHANToM devices, one for each hand. Although technically spoken, a distributed solution, spreading the haptic load onto a network, appeared to work fine, this solution lacked the feeling of proprioception.

In section 7.4 we proposed the 'Object in Hand' metaphor, in which the proprioceptive knowledge of the non-dominant hand is used to activate and bring a menu in position. Subsequently we explained how this solution can be used to 'grab' an object from the world and bring it to a central position, where it can be manipulated or inspected. Although an experiment turned out that this solution has its value, the 'object accessing problem', stated in the introduction, now has been changed for an 'object selection problem'.

In section 7.6, we elaborated on three selection metaphors we already introduced in chapter 3, and evaluated how they perform using the non-dominant hand. From this experiment we could conclude that the aperture selection performs best, and that there is little difference between the dominant and the

non-dominant hand. The result allowed us to integrate the aperture selection, performed by the non-dominant hand, into the 'Object in Hand' metaphor.

We ended this chapter by an experiment to evaluate this combined solution. We found that there is no significant degradation of the solution, neither in the selection phase, nor in the manipulation phase, as the scene runs more complex. Although, the latter experiment thought us some issues which may be solved in the near future.

In chapter 12, we will indicate how the currently proposed solutions can be further improved in the future. In part III of this thesis, we will first elaborate on the more technical aspects, which formed a basis for the solutions and experiments explained in the previous chapters, giving a designer's perspective on multimodal interaction in a 3D environment.

Part III

Technical Aspects

Chapter 8

Haptic Rendering

Contents

8.1 Introduction

In the previous chapters, we have discussed our research towards a cooperative bimanual interaction technique, using the sense of proprioception and force feedback, which may improve the interaction in a 3D environment. It may be clear however that the development of such solutions are the result of a long and expensive development process. In this part of the thesis, we

will go in detail into the more technical aspects of the implementation and how, from a designer's perspective, this process can be simplified. In this chapter, we will discuss some topics on the performance of haptic rendering. First we measure the performance of two PHANToMs connected to the same machine [De Boeck et al., 2002a] as the result of this experiment may motivate our solution to distribute the haptic calculations on a network. As this was a first experiment which needed to evaluate the performance of haptic algorithms in practice, the next chapter discusses a more formal way to evaluate haptic algorithms [Raymaekers et al., 2005a][De Boeck et al., 2005b].

In chapter 9, we explain some issues regarding our research framework, on which the experiments were built. The chapter also illustrates how this framework may be used to distribute the haptic simulation across several computers on a network, as already mentioned in section 7.3. Chapter 10 explains the design decisions made to make abstraction from the physical devices and to interpret NiMMiT diagrams at runtime.

8.2 Haptic Algorithms and Related Work

It is clear that haptic interaction has been a growing field within the research on interactive 3D environments. A problem with haptic feedback is the fact that our haptic sense is relatively sensitive compared to our other senses (see table 5.1 for some relevant numbers). Besides the performance requirements on the hardware in terms of inertia, peak force or acceleration [Hayward and Astley, 1996], this also lays a constraint on the software. Forces must be recalculated at least at 1Khz, implying that haptic algorithms, responsible for calculating those forces, typically are executed within a dedicated thread (the 'haptic loop') which may take less than 0.9ms per time frame. When rendering very stiff objects, an even higher update rate of 5-10kHz is desired [Kabeláč, 2000]. This is a very demanding requirement if we compare this to the 25Hz update rate of the graphical output.

We will first define what we mean by a haptic algorithm. This is not a formal characterisation; its purpose is to clarify our working definition in order to avoid confusion. A more formal explanation of force feedback algorithms can be found in [Ruspini, 1999].

Definition 8.1 *A* geometric haptic algorithm *a is a two-fold algorithm. Its input is the current position* $\overrightarrow{p}$ *and velocity* $\overrightarrow{v}$ *of the pointer, as defined by the force feedback device and a virtual object o, which has to be rendered. The*

first part, denoted $coll(a, o, \overrightarrow{p})$ *is a collision-detection step, which calculates whether the pointer position is located inside the object. The second step, denoted by* $render(a, o, \overrightarrow{p}, \overrightarrow{v})$ *calculates the surface contact point (SCP) and the force that should be exerted by the force feedback device. As depicted in Fig. 8.1 a spring-damper system is placed between the 'real' pointer position and the SCP. The output of the haptic algorithm is a force* $\overrightarrow{F}$*, which is the result of this spring-damper system.*

In this thesis, we refer to geometric rendering, not volumetric rendering, as most volumetric rendering techniques do not make use of a SCP (see [Avila, 1999] for more information on volumetric rendering). Also, some haptic rendering algorithms use a SCP only in implicit calculations (e.g. the e-Touch library [Novint, 2001]). However, the principles stay the same in these cases. For the brevity of our definitions and explanations, we will use definition 8.1 further as a basis in this thesis.

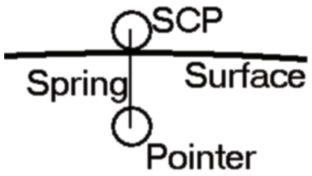


Figure 8.1: Surface Contact Point

Although little research into the correctness of haptic algorithms has been performed, it remains clear that research to both the performance as the correctness of a haptic algorithm plays an important role in the development of better and more realistic simulations. Recently, some empirical methods were developed for the evaluation of haptic applications [Kirkpatrick and Douglas, 2002] or virtual environments in general [Sutcliffe and Gault, 2004]. In the past, several attempts have been made to quantify the benefits of a haptic algorithm. Theoretical approaches tried to quantify the time complexity of different algorithms (e.g. [Raymaekers et al., 2001a]). Although this leads to a better understanding of the algorithms' performance, it does not allow for the comparison of two algorithms with the same time complexity. Furthermore, due to the behaviour of the end user, some optimisations are difficult to predict. Thus, these theoretical findings should be complemented with real-life measurements in order to have an idea of the exact results.

As an example of an empirical approach, we mention the work of Acosta and

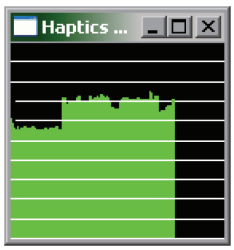


Figure 8.2: GHOST haptic load tool

Temkin, who compared the performance of different versions of the GHOST haptic API [Acosta and Temkin, 2001]. With a 'Load until fail' method, they looked at the implementation of polygonal meshes by loading objects with as many triangles as possible, until the system was not able to process the information within the 1Khz restriction. In a similar manner, they measured the performance of the haptic scene graph implementation. Although, it does not take the performance at a lower load into account, this approach is suited to test the limits of an implementation, which is a valuable metric. We will partly adopt this methodology in section 8.3 when we measure the increase of the haptic load[1] when a second PHANToM device is present.

A third possibility of measuring an algorithm's performance is to use a tool which measures the haptic load. The GHOST haptic API provides a graphical tool which displays the haptic load using 10% intervals, as depicted in Fig. 8.2. This technique was used to compare the polygon mesh implementation of GHOST and e-Touch [Anderson and Brown, 2001]. We will also use this approach in section 8.3. Certainly, this gives a good insight of the algorithms' performance. In our opinion, however, this does not provide accurate numerical results which can be validated.

In the next section, we use the 'Load until fail' method together with the haptic load tool in order to measure how the GHOST API behaves when a second PHANToM device is connected to the same computer. After that, in section 8.4, we elaborate on a more formal method to compare haptic algorithms, allowing to do statistical processing on the results. The latter method, however, requires a haptic library, which is open enough to add some extra code.

[1] The haptic load is the processor time spent in the haptic loop to calculate the forces

8.3 PHANToM Haptic Load

8.3.1 Introduction and Related Work

In this section, we elaborate on two experiments. In a first experiment, we measure the haptic load in a scene that contains one single object with increasing complexity. This object has been created by subdividing either a tetrahedron or a cube using the Loop subdivision scheme described in [Loop, 1987] and in [Raymaekers et al., 2001a]. Each subdivision level is represented in the haptic scene graph by an instance of a haptic polymesh. In order to test if the haptic load depends on the object's geometry, we also have used more natural models with a high number of polygons such as a rabbit and a fish.

In a second test, we measured the haptic load in a scene with an increasing number of objects. These objects are positioned in a 3D matrix (as in [Acosta and Temkin, 2001]) growing in each direction. The objects do not intersect each other's bounding box and they also varied in complexity, using the same techniques as in the first experiment.

Both experiments have been conducted with a single and a dual PHANToM setup. The criterion measured in the two experiments is the haptic load, as measured with Sensable's Haptic Load (HLOAD) tool, until the 1kHz requirement can not be satisfied anymore.

8.3.2 Setup

The computer used in our experiments was a Pentium III 600 MHz, 256 MB RAM, running Windows NT SP6. Due to an incompatibility between the PHANToM PCI interface card, the AGP adapter and the computer's motherboard, unfortunately, we were forced to conduct our tests with a poor video card. However, since the GHOST thread is a high priority thread, we believe this has little or no effect on the haptic load measured. Our tests have been conducted on Windows NT, because we did not succeed in connecting two PHANToM devices on Windows 2000 or XP. Our assumption that the choice of our hardware and OS does not significantly influence our results, are confirmed by comparing our single PHANToM test results on a Pentium III 850 MHz and a dual Pentium III 800 MHz running Windows 2000.

Table 8.1: Haptic load in percentage in a single (1P) and double (2P) PHANToM Setup with complex objects.

			1 Phantom			2 Phantoms		
			N Contact	Contact	Contact	N contact	1 Contact	2 Contacts
Subdivision		#tri	*1P/0C*	*1P/1C*	& Moving	*2P/0C*	*2P/1C*	*2P/2C*
Tetraedron	level 0	4	<10	>10		10	10< - <20	<20
Cube	level 0	12	<10	>10		>10	10< - <20	<20
Tetraedron	level 1	16	<10	>10		>10	10< - <20	<20
Cube	level 1	48	<10	>10		>10	10< - <20	<20
Tetraedron	level 2	64	<10	>10		>10	10< - <20	<20
Cube	level 2	192	<10	>10		>10	10< - <20	20
Tetraedron	level 3	256	<10	>10		>10	10< - <20	20
Cube	level 3	768	<10	>10		>10	10< - <20	>20
Tetraedron	level 4	1024	<10	>10		>10	10< - <20	>20
Cube	level 4	3072	<10	>10		>10	10< - <20	>20
Tetraedron	level 5	4096	<10	>10		>10	10< - <20	>20
Cube	level 5	12288	<10	>10		>10	>20	>30
Tetraedron	level 6	16384	<10	<20		>10	>20	>30
Cube	level 6	49152	<10	20	30	>20	40	<60
Tetraedron	level 7	65536	<10	30	40	<20	40	>70
bunny		69451	<10	50	60	<20	70	quit
Fish		100480	<10	30	90	20	Unstable	quit
rabbit		134074	<10	80	quit	<20	quit	quit
Thetraedron	level8	262144	<10	70	quit	quit	quit	quit

8.3.3 Results

Experiment 1: Objects with increasing complexity

Table 8.1 summarises the results of the first experiment. The values indicate a percentage of the haptic load of the simulation: when indicating 10% it means that this amount of the total haptic loop (of 1ms) has been spent for the calculations. When looking at the first column where no contact is made, one can see a quite constant haptic load both in the single PHANToM (1P/0C) as in the dual PHANToM setup (2P/0C). When touching the object (without moving the surface contact point over the surface) (1P/1C) the haptic load starts increasing from a shape with 160,000 triangles. This is consistent with the values reported in [Anderson and Brown, 2001]. As can be seen in the next column, the haptic load augments when the surface contact point moves over the object's surface. This causes the GHOST-thread to quit with the most complex models in our experiment. With the dual PHANToM condition, the haptic load again is quite constant when no contact is available (2P/0C), but slightly rises when increasing the object's complexity. When one PHANToM touches the object (2P/1C), the haptic load is somewhat higher compared with the single PHANoM setup. The table here suggests a higher increase for the more complex models. The second surface contact point introduces yet another increase, in such that the total haptic load is roughly 50% more than in (1P/1C). The results of those experiments are graphically depicted in figure 8.3

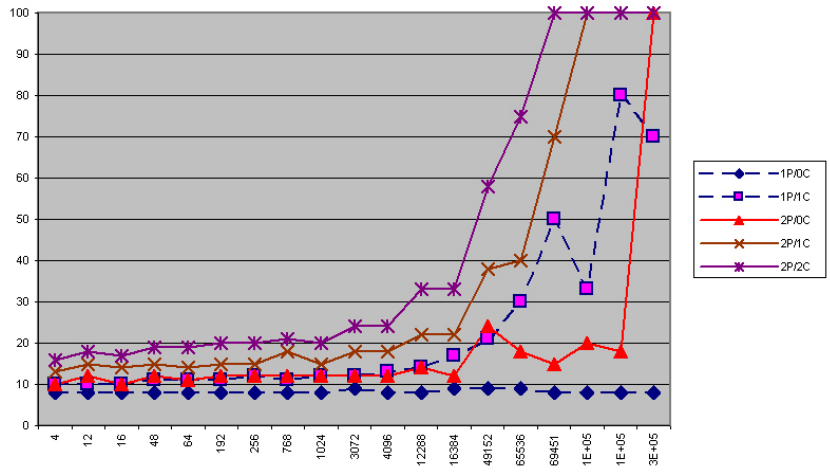


Figure 8.3: Graph of the haptic load in a single and double PHANToM Setup with complex objects.

Table 8.2: Haptic load with multiple objects. (subdivision level 0)

Tetrahedron Subdivision Level 0 (4 triangles)					
	One PHANToM		Two PHANToMs		
# objects	No Contact	1 Contact	No Contact	1 Contact	2 Contacts
8	10	<20	>20	>30	>40
27	<20	>30	30	<50	>60
64	>30	<70	<60	quit	quit
125	<50	quit	quit	quit	quit
216	quit	quit	quit	quit	quit

Experiment 2: Scenes with increasing complexity

Tables 8.2, 8.3 and 8.4 show the results of the haptic load in a scene with an increasing number of objects. Each table displays the same number of objects, but with more triangles per object. All the objects are placed in a grid so that their bounding boxes do not overlap. A single PHANToM setup can fully support a scene with up to 64 tetrahedrons, while the dual PHANToM setup has the same load when simulating only 27 objects. The results of table 8.4 show that the haptic loop with 2 PHANToMs quits when touching one of the level 4 subdivision objects in the scene, while the single PHANToM setup still supports 64 objects.

Table 8.3: Haptic load with multiple objects. (subdivision level 2)

Tetrahedron Subdivision Level 2 (64 triangles)					
	One PHANToM		Two PHANToMs		
# objects	**No Contact**	**1 Contact**	**No Contact**	**1 Contact**	**2 Contacts**
8	>10	20	<20	>20	30< - <40
27	<20	30< - <40	>30	>50	<70
64	>30	50< - <60	<70	quit	quit
125	<70	quit	quit	quit	quit
216	quit	quit	quit	quit	quit

Table 8.4: Haptic load with multiple objects. (subdivision level 4)

Tetrahedron Subdivision Level 4 (1024 triangles)					
	One PHANToM		Two PHANToMs		
# objects	**No Contact**	**1 Contact**	**No Contact**	**1 Contact**	**2 Contacts**
8	10	<20	70	quit	quit
27	20	>40	quit	quit	quit
64	<40	60	quit	quit	quit
125	<70	quit	quit	quit	quit
216	quit	quit	quit	quit	quit

8.3.4 Discussion

Results

Our values in the single PHANToM case correspond to the findings of [Acosta and Temkin, 2001] and [Anderson and Brown, 2001]. Although, Acosta et al. can support up to 600 cubes, while our simulation already quits at 125 objects. This may be the result of Acosta's implementation using the standard haptic cube from the API, which is simpler, more efficient, but less general than the haptic polymesh, used in our experiment. From the first experiment, we can conclude that the haptic load in a dual PHANToM setup has been increased, compared to the single PHANToM setup. As long as the object is relatively simple (up to 16,000 triangles) the haptic load keeps below 30%, which results in a stable simulation. Although the haptic load tool, which was the only tool we had to measure, is not the most accurate tool one can imagine, we roughly can say that the second PHANToM increases the haptic load with about 50%. This can be confirmed by comparing the number of triangles in the most complex, but stable simulation in the first condition (fish with 100.480 triangles) with the maximum number of triangles in a stable dual PHANToM simulation (tetrahedron level 7 with 65.536 triangles). Even more pronounced is the increase of the haptic load in a scene with multiple

objects. If we compare a 'single contact with one PHANToM' (1P/1C) with a 'double contact with two PHANToMs' (2P/2C), we see that the haptic load almost doubles. This makes that complex scenes, which can be run with one PHANToM, are not supported by a dual PHANToM setup.

8.3.5 Other findings

During the course of our experiments, we have encountered a number of interesting situations. Some of them appear to be obvious, but we believe they can give the programmer a better understanding of optimising a more complex scene.

- When starting the haptic loop, very often a 'haptic load spike' is encountered. This is why heavy models often crash at the beginning of the simulation. However if a complex simulation accidentally 'survives' a startup, the simulation seldom is completely stable. Most of the time the haptic thread quits when interacting in the scene.

- When exploring the 'natural' objects (fish, rabbit), the haptic load was higher when approaching a more complex region with lots of small triangles. This is not surprisingly, knowing that the meshes are optimised using spational partitioning.

- When sliding the PHANToM over an object, this increases the haptic load, compared to a static contact. On the other hand, a static contact, which touches more than one triangle (e.g. in a corner), requires more processing time.

- Unfortunately, due to some known compatibility issues in the API, we could not make Windows 2000 to work with the two PHANToMs. But, for reference purposes, we have conducted some of the single PHANToM tests on a Windows 2000 PC, as well. This allows us to conclude that there is little or no difference between the results of these tests compared to the tests running on Windows NT.

- At first sight, the dual processor seems to perform better in starting-up a very complex scene: scenes that quit at start-up on a single processor machine, do run on the dual processor computer. When interacting in those scenes, however, the results are very similar to a single processor machine.

- On a dual processor computer other tasks will slow down less when executing a heavy GHOST thread. For instance, when running complex scenes, the haptic loop will slow down the graphics on a single processor computer, which is not true on a dual processor machine. This is quite obvious because the ghost thread in some cases can take up to about 99% of one processor's time, which is only 50% of the total processing power of a dual machine.

8.3.6 Conclusions

From the experiments described in this section, we have learned that adding a second PHANToM to the same computer in order to support a multi-contact haptic simulation, significantly increases the haptic load on that computer, and hence limits the complexity of the scene to be rendered. Despite some possible incompatibility issues we encountered, a single machine solution is by far the most easy way to implement multiple contact points. More complex scenes, however, will need another approach. In section 7.3, we propose an architecture which allows to distribute the haptic load onto a network, supporting several computers, each driving one haptic device.

8.4 Evaluating the Performance of a Haptic Algorithm

8.4.1 Introduction and Related Work

In the previous section, we used the 'Load until Fail' method, together with the Haptic load tool in order to evaluate the increase of the haptic load, when a second PHANToM is connected. This approach gives an impression of the behaviour of an algorithm under the tested circumstances. However, those results are far from accurate as it does not provide us with exact numerical results. Moreover, this approach does not relay on the same data provided to all algorithms or situations, as the results are measured while interacting in real-time with the environment. Hence, unintentionally, the comparison is not conducted in an equal manner. Even more, Anderson and Brown [Anderson and Brown, 2001] for instance, compare two object rendering algorithms which are implemented in two different APIs, unintentionally measuring scene graph performance, and possible other artifacts, as well.

In this section, we address this problem by presenting the same recorded data to different algorithms. In an interactive session, in which users explore a virtual object with a haptic device (in our case, a PHANToM device), the device's position and velocity are recorded for each loop. This data is then passed on as input for the other haptic algorithms. Different variables such as the time needed to execute one haptic loop, are recorded and compared. This approach is very similar to a parallel research of Ruffaldi et al. [Ruffaldi et al., 2006]. In their work, however, they record 'real world' paths and forces (instead of virtual), and statistically compare the results of the haptic algorithm under test to these real world values.

In the following paragraphs, we will first discuss our evaluation methodology, both from a theoretical and a practical point of view. In section 8.4.3 we give an example of an empirical evaluation in order to assess the validity of our approach, and finally, the results are discussed in section 8.4.4.

8.4.2 Evaluation Methodology

Our evaluation methodology is an empirical implementation which relies on a number of definitions of algorithm performance. We will first expand upon these definitions before going into detail on their practical use.

Working Definitions

First of all, as a basis for our working definitions, we refer to definition 8.1 *'(geometric) Haptic Algorithms'*.

We will denote the set of all possible haptic algorithms by $\mathcal{A}$.

Definition 8.2 The *object space* $\mathcal{O}(a)$ of a haptic algorithm a is the set of virtual objects on which the algorithm can be applied.

Not all objects belong to the object space of an haptic algorithm, as these cannot be applied on every possible object (for instance a rigid-body algorithm cannot perform soft-body computations). Remark, however, that an algorithm's object space contains an infinite number of virtual objects.

Definition 8.3 A *path* is a collection of points in space which are followed by a user's pointer containing the velocities of the pointer in those points, as well.

Definition 8.4 The *length* of a path p, denoted by $length(p)$, is the duration in time, that the users need to explore the path. The position on the path on a certain point in time t ($t \in [0, length(p)]$) is denoted by $\overrightarrow{p}_t$; its velocity by $\overrightarrow{v}_t$.

When users explore a certain object, there are many possible paths they can follow. For instance a solid rigid body cannot be penetrated (although practical implementations of course allow a slight penetration). On the other hand, a deformable object cannot be deformed in all possible manners: a virtual football can be slightly squeezed, but it is impossible to reach its centre.

All possible paths that a user can follow when working in an empty virtual environment is called the path set $\mathcal{P}$.

Definition 8.5 The *path space* $\mathcal{P}(o) \subset \mathcal{P}$ of a virtual object o is the set of paths that can be followed by a user when exploring that object and thus touches the object.

An object's path space contains an infinite number of paths. Furthermore, a path can belong to the path spaces of an infinite number of objects.

Equivalence of Algorithms

Before defining the equivalence of algorithms, we first introduce the equivalence of the different parts of the algorithms.

Definition 8.6 *x The* collision-detection *steps of two haptic algorithms a and b are* equal *for a certain object o and a point in space $\overrightarrow{p}$ if either both collision-detection steps return true or they both return false. We denote this by*

$coll(a, o, \overrightarrow{p}) \simeq coll(b, o, \overrightarrow{p})$.

Definition 8.7 The *render* steps of two haptic algorithms a and b are *equivalent* for a certain object o, a point in space $\overrightarrow{p}$ and a velocity $\overrightarrow{v}$ if the difference between the *SCPs* returned by both render steps is smaller than the just noticeable difference[2] (JND). We denote this by

$render(a, o, \overrightarrow{p}, \overrightarrow{v}) \cong render(b, o, \overrightarrow{p}, \overrightarrow{v})$.

[2]The just noticeable difference is the smallest change in pressure, position, ... that can be detected by a human and depends on the body area where the stimulus is applied [Burdea, 1996]. As an example, one cannot tell the difference between two orientations of one's wrist if they are less than 2.5° apart.

We use the term 'equivalent' instead of 'equal' in definition 8.7 because two equivalent render steps can return a different force.

Definition 8.8 The *render* steps of two haptic algorithms a and b are *equal* for a certain object o, a point in space $\overrightarrow{p}$ and a velocity $\overrightarrow{v}$ if these render steps are *equivalent* and if the difference between the *forces* returned by both steps is smaller than the JND. We denote this by

$render(a, o, \overrightarrow{p}, \overrightarrow{v}) \doteq render(b, o, \overrightarrow{p}, \overrightarrow{v})$.

Using definitions 8.6 through 8.8, we can now define the equivalence of haptic algorithms.

Definition 8.9 We call two haptic algorithms *collision equivalent* if their collision-detection steps are equal in all cases:

$\forall a, b \in \mathcal{A} : (\mathcal{O}(a) = \mathcal{O}(b)) \wedge (\forall o \in \mathcal{O}(a), \forall p \in \mathcal{P}(o), \forall t \in [0, length(p)] : coll(a, o, \overrightarrow{p}_t) \simeq coll(b, o, \overrightarrow{p}_t)) \Rightarrow a \simeq b$.

Definition 8.10 Two haptic algorithms are *render-equivalent* if their render steps are equivalent in all cases:

$\forall a, b \in \mathcal{A} : (\mathcal{O}(a) = \mathcal{O}(b)) \wedge (\forall o \in \mathcal{O}(a), \forall p \in \mathcal{P}(o), \forall t \in [0, length(p)] : render(a, o, \overrightarrow{p}_t, \overrightarrow{v}_t) \cong render(b, o, \overrightarrow{p}_t, \overrightarrow{v}_t)) \Rightarrow a \cong b$.

Definition 8.11 Finally, two haptic algorithms are *equal* if their render steps are equal in all cases:

$\forall a, b \in \mathcal{A} : (\mathcal{O}(a) = \mathcal{O}(b)) \wedge (\forall o \in \mathcal{O}(a), \forall p \in \mathcal{P}(o), \forall t \in [0, length(p)] : render(a, o, \overrightarrow{p}_t, \overrightarrow{v}_t) \doteq render(b, o, \overrightarrow{p}_t, \overrightarrow{v}_t)) \Rightarrow a \doteq b$.

We define the three levels in definitions 8.9 through 8.11 because different algorithms can have different purposes. For instance, a new algorithm can be created in order to be faster. In that case, the new algorithm should be equal to existing implementations.

It is also possible that a new algorithm modifies the force vector in order to implement a haptic texture [McGee et al., 2001]. This algorithm can be render-equivalent to existing algorithms.

Finally, a new algorithm could be created which rounds certain edges by modifying the SCP. This algorithm can be collision equivalent to existing algorithms.

Correctness of Algorithms

Comparing a new algorithm to all existing algorithms that share the same object space is very time consuming. We therefore propose that for each possible rendering problem a reference algorithm is defined. Such a reference algorithm is generally accepted to solidly solve the rendering problem. At this moment, no reference algorithms are defined, but one can use the widely accepted implementations that are used in SDKs, such as GHOST [SensAble, 2001].

Definition 8.12 The set of all possible *reference algorithms* is denoted by $\mathcal{R}$, where $\mathcal{R} \subset \mathcal{A}$.

We can now define if an algorithm is correct with respect to a certain reference algorithm and a certain operator.

Definition 8.13 $\forall a \in \mathcal{A} \backslash \mathcal{R}, \forall r \in \mathcal{R}, \forall op \in \{\simeq, \cong, \doteq\} : correct_{r,op}(a) \Leftrightarrow a\ op\ r$.

The operator used in the definition depends on the problem that has to be solved. Collision equivalence is the minimal requirement in order to guarantee that a comparison of algorithm's performance, as defined in the next subsection, is possible.

Algorithm Performance

Based on the definitions of section 8.4.2, we can now define the mutual performance of two algorithms. We will first define how the performance of one algorithm can be measured in order to define the comparison of two algorithms.

Definition 8.14 The *rendering time* of an algorithm a for a virtual object o, a point in space $\overrightarrow{p}$ and a velocity $\overrightarrow{v}$, denoted by $time(a, o, \overrightarrow{p}, \overrightarrow{v})$ is the time needed to calculate $coll(a, o, \overrightarrow{p})$ and $render(a, o, \overrightarrow{p}, \overrightarrow{v})$.

One can now state that an algorithm is better than another algorithm if they are both correct (we do not take incorrect algorithms into account) and the former's rendering time is smaller than the latter's.

Definition 8.15 $\forall a, b \in \mathcal{A}, \forall op \in \{\simeq, \cong, \doteq\} : a\ op\ b \wedge \forall o \in \mathcal{O}(a) : \forall p \in \mathcal{P}(o), \forall t \in [0, length(p)] : time(a, o, \overrightarrow{p}_t, \overrightarrow{v}_t) \leq time(b, o, \overrightarrow{p}_t, \overrightarrow{v}_t) \Rightarrow a \geq_{op} b$

Or it is strictly better if:

Definition 8.16 $\forall a, b \in \mathcal{A}, \forall op \in \{\simeq, \cong, \doteq\} : a \geq_{op} b \wedge \exists o \in \mathcal{O}(a), \exists p \in \mathcal{P}(o), \exists t \in [0, length(p)] : time(a, o, \overrightarrow{p}_t, \overrightarrow{v}_t) < time(b, o, \overrightarrow{p}_t, \overrightarrow{v}_t) \Rightarrow a >_{op} b$

Of course, an algorithm that is in some rare case slower than another algorithm and has an overall better performance is still better than the latter algorithm. This is not reflected in definitions 8.15 and 8.16. We therefore take this fact into account in our last definition.

Definition 8.17 An algorithm a has *a better overall performance* than algorithm b for an operator $op \in \{\simeq, \cong, \doteq\}$ if $a\ op\ b$ holds true and if the rendering time of a is significantly less than the rendering time of b (when comparing both rendering times using a relevant statistical technique). We denote this by $a \succ_{op} b$.

As a consequence of this definition, the following always holds true:

$$a >_{op} b \Rightarrow a \succ_{op} b$$

Practical Implications

When bringing the definitions of section 8.4.2 into practice, a number of issues arise. Although we define a set of reference algorithms that are assumed to be correct, it is very hard to prove the correctness of an algorithm without having a basis to compare to. A reference implementation may not even exist yet. Indeed, in order to verify if an algorithm is correct, one must evaluate an infinite number of situations (each object space contains an infinite number of objects and each path space contains an infinite number of paths).

Secondly, if we assume to have a reference algorithm, we have to compare an infinite number of objects and paths to prove the correctness of another algorithm.

In order to evaluate a new algorithm, as mentioned before, we propose to use an empirical method and use statistical techniques to draw conclusions. As a reference algorithm, we therefore propose to use an algorithm that already has proven its 'correctness' in practice.

Next, a number of reference objects (n_o), with different shapes and different numbers of polygons have to be chosen. We recommend to choose both concave

and convex objects with a varying number of triangles. For each object, a number of paths can be generated by letting a number of users (n_u) explore the object for a certain amount of time (t_e), expressed in seconds. In order to measure differences in the user's behaviour, we suggest to choose several differently skilled subjects (such as novice, experienced and expert users).

We now sample the paths of all users using the reference algorithm and store the pointer positions and velocities. If we assume f_{sr} as the update rate of the haptic loop (in Hz), this makes a total of

$$n_o \times n_u \times t_e \times f_{sr}$$

samples per algorithm that need to be tested. Next, the recorded samples are played back using another algorithm and the results are logged. For each algorithm, one can now statistically compare the execution time (e.g. using a student t-test), the result of the collision step, the surface contact point and the force vector. This allows us to make judgements about the performance and the correctness as described in 8.4.2.

For instance if we define a test with 5 objects ($n_o = 5$), explored by 6 users ($n_u = 6$) for a time of 40 seconds ($t_e = 40$) and if we assume a sample rate of 1000Hz ($f_{sr} = 1000$), we will end up with 1.200.000 samples per experiment. Although the researcher is free to choose the size of the experiment, we have to take the middle course between a representative set of objects and users, and a manageable number of sample points. In this context Raymaekers et al. conducted an additional experiment in order to make recommendation about the desired complexity of the tested objects, as well as the required experience of the users. [Raymaekers and Coninx, 2006]

To sum up, we propose a four-fold solution in order to test an haptic algorithm:

1. Choose a reference algorithm and n_o objects;
2. Let n_u users explore the objects and record the paths by their positions and velocities during a given time t_e;
3. Apply the algorithms that are to be tested on the acquired data set;
4. Use statistical techniques to sample the differences.

8.4.3 Example Evaluation

To assess the proposed technique, a comparison between two algorithms has been conducted in practice. As GHOST is currently integrated in our research framework [De Boeck et al., 2003a], we have chosen to record the playback data using the GHOST API. In order to allow the recording of our samples, GHOST has been extended by deriving a new class from the polymesh implementation class. The derived class invokes a first callback function before starting the collision step and a second callback after the SCP calculation. The callback functions write the position and velocity of the virtual pointer to one file and the results of the haptic calculation to another file.

As GHOST does not allow to insert custom code for measuring purposes at any particular place, and as it is nearly impossible to know what calculations exactly are performed in the executed interval (scene graph calculations, object rendering calculations, or even other calculations), the logged results from GHOST were not formally evaluated. Instead, they were used as a basis for the evaluation of two test algorithms.

Two test algorithms were developed within HAL, a **ha**ptic **l**ibrary we developed for research purposes [Raymaekers et al., 2005a]: an algorithm, based on the God-object algorithm, a standard SCP-based geometric algorithm defined by [Zilles and Salisbury, 1995], and a second algorithm, which optimises the previous by implementing spational partitioning using an octree [Anderson and Brown, 2001].

Two specialised classes were written for HAL. A pseudo haptic device reads the positions and velocities from file and acts as if it was an haptic device using the recorded data. A scene graph with logging features measures the time that an algorithm needs for haptic calculations and saves the result of the algorithm and the time that was needed. The measurements were made using the Windows high-performance timer. Although Microsoft Windows is not a real-time operating system, we made the calculations more precise by executing the test algorithms within a high-priority thread, as a regular haptic device would do. Just before the execution of a new loop, the measurement scene graph lets the other processes execute by releasing its time slice. This decreases the chance that a context switch occurs in the middle of the haptic calculations. Finally, the pseudo-haptic thread is bound to one processor in order to avoid problems with the high-performance timer[3].

[3]The high-performance timer can give results that deviate slightly when a thread migrates from one processor to another.

As this is a proof of concept validation of our measurement method, we did not perform a full evaluation of the test algorithms. This means we didn't explicitly choose a reference algorithm from which we can be 100% sure it is correct. We also chose the values of our parameters to be relatively small to end up with a manageable amount of data. We have tested three objects ($n_o = 3$), a cube, consisting of 12 triangles, a sphere, consisting of 182 triangles and a sphere consisting of 562 triangles. Only one expert user ($n_u = 1$) tested the objects during 40 seconds ($t_e = 40s$). The haptic update rate was 1000Hz ($f_{sr} = 1000$), resulting in a set of 120.000 samples per algorithm. All the values were stored in a SQL database, which allows us to easily select and combine results and export parts to statistical applications.

8.4.4 Discussion

This section discusses the results from the example evaluation. First, we will compare the results from both algorithms. Next, we will generalise our results and explain some other benefits of our evaluation method.

Comparison

Let us assume the correctness of algorithm 1, which is based on the God-object algorithm described in [Zilles and Salisbury, 1995]. First the collision equivalence of both algorithms has to be verified. If two algorithms are not collision equivalent, one cannot unambiguously draw conclusions from this test. Collision equivalency can be very easily checked by performing a query that only selects those samples in which the collision result of both algorithms was different. From our values we can even conclude the equality of both algorithms, which could be expected since algorithm 2 is an optimised version of algorithm 1. Note that to conclude the correctness of an algorithm, we don't require render equivalence or equality.

If we compare the calculation times for the samples in which both algorithms have collision, and the times of the samples in which no collision occurs, and we compare both algorithms using one-way ANOVA, we can conclude a significant improvement for the optimised algorithm ($p<0.001$). Since we could predict this result in advance, the proposed approach shows a statistically founded conclusion.

Moreover, if the test has been conducted by a variety of objects or a variety of skilled users, the database allows us to select a subset of samples and draw

conclusions about special conditions. In our example we have sampled two spheres, one with 182 and one with 562 polygons. If we compare the improvement of calculation time of the low-resolution object, with the improvement of the high-resolution object, we can conclude that algorithm 2 has a higher improvement for complex objects. This result can also be checked against the theoretical time complexity of the algorithm. For illustration, the average calculation times for calculating the collision detection and the SCP is summarised in table 8.5. As one could expect, the calculation time increases with the number of triangles in algorithm 1, while the octree of algorithm 2 causes the calculation time to stay near-constant.

Table 8.5: Average calculation times for the spheres

Triangles	Algorithm 1	Algorithm 2
182	0.01029ms	0.00746ms
562	0.03175ms	0.00783ms

Generalisation

Although our proposed measurement method has been tested with an early version of our haptic library, we are convinced that the results are far more general. We want to stress that this test can be implemented in any haptic API that is open enough to implement the necessary recording or play-back features.

To record values of a path, it is sufficient to read the device's position, velocity and collision result at the given sample rate. All current available API's support this feature. To measure the calculation time, it is necessary to capture the high-performance timer before and after the calculation. In the GHOST API we can do this by deriving a new object. It is impossible to measure different scene graph algorithms as those timings don't take any scene graph calculation into account. Other API's such as e-Touch, which are open source don't suffer from this problem since their code can be extended at any point.

To playback values, a new device has to be created in order to read the recorded paths from file. For this feature we need an API that not only allows multiple devices, but also allows the developer to implement a pseudo device. This requirement makes it impossible for new algorithms to be tested in an implementation within the GHOST API. New algorithms that have to be tested

therefore must be implemented in an API that is open enough to support these features.

Other Benefits

Since our method allows us to record a path and replay exactly the same data with other algorithms, and since we are recording the results to a database, we are able to quickly find those samples where erroneous differences between the algorithms occur. Using the same recorded paths and identical values, results in a deterministic system which makes it easier for the programmer to reproduce the same errors instead of searching for the same coincidental situation. By debugging the algorithm with just a subset of the recorded path, the error can be traced line by line. In preparation of this experiment, we have used this method to eliminate some implementation errors in the collision detection algorithm and in the octree algorithm.

8.5 Summary

In this chapter we went into the evaluation of haptic algorithms. In a first section, we investigated the additional computational load when connecting two PHANToM devices to the same computer, as we considered this approach for our two-handed haptic setup, described in chapter 7. We found that the increase is quite significant, resulting in less, or less complex objects that can be supported within the world. In section 9.3 we will show how this limitation can be solved by distributing the haptic devices across a network.

As the applied method for this evaluation did not provide us with accurate numerical data, section 8.4 proposed a more systematic evaluation method. We proposed a formal evaluation method that can be applied in order to test new algorithms for speed and correctness. This approach has been illustrated by a concrete example.

In the next chapters, we will elaborate on some design and implementation issues which were necessary for this thesis to support the solutions proposed in section II.

Chapter 9

A Research Framework to Support Experiments

Contents

9.1 Introduction

In this chapter, we describe a software framework, built in our lab, which forms the basis to support the research presented in this thesis. We will first explain the main building blocks of the framework. Next we show in detail how this framework can be applied to provide a two-handed haptic experience with two PHANToMs, by distributing the haptic calculations across a network.

9.2 Multithreaded Framework

9.2.1 Situation

It may be clear that writing interfaces for applications that support 3D multimodal interaction is a long and expensive process. To shorten the development cycle of our experiments (such as described in Part II of this thesis), for several years, a code framework has been built in our lab. This framework, called 'VRment', especially fits the specific needs for our experiments. Moreover, this framework has been applied and extended as an essential part of the VR-DeMo [EDM-WISE, 2004] project, a project that investigates 'if' and 'how' a model-based approach[1] for the development of applications can be applied to 3D environments.

Several tools and frameworks currently exist that may assist a designer or developer to create interactive 3D environments. Although products such as VRJuggler [Iowa State University, 2006], 3D Game Studio [3DGameStudio community, 2006] or Virtools [Virtools inc, 2006] may be suitable for the majority of practical applications, none of these tools fully supports our multimodal research requirements (haptics, speech, PSD, ...), network support or hybrid user interface elements. Therefore, instead of adopting an entire solution, 'VRment' relies on existing APIs and packages. Some relevant APIs include:

- Xerces for reading XML files
- VRPN [Seeger et al., 1997][Tailor II, 2006] for input device abstraction
- GHOST [SensAble, 2001] or HAL [De Boeck et al., 2005b] for force feedback
- SOLID [van den Bergen, 2001] for collision detection
- Microsoft Speech API for speech recognition
- OGRE3D [Ogre Community, 2006] for visual rendering

[1] Model based development is a topical subject in the domain of dialog based and multi-device interfaces [Coninx et al., 2003]. The aim is to 'describe' the interface or application by means of a model instead of 'implementing' it with programming code

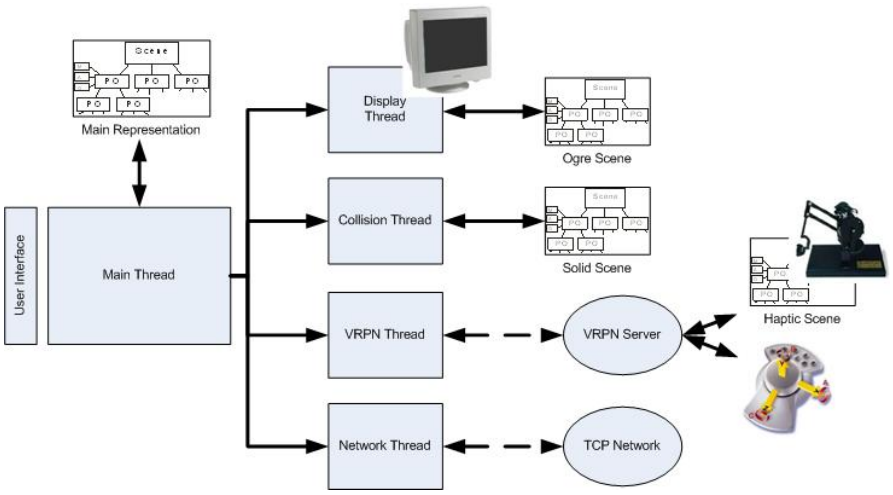


Figure 9.1: Threads in VRment

The box at the bottom of figure 9.2 shows the entire list of APIs.

In the next section, we will go into more detail on the general built-up of the framework. This background will be used throughout the following sections in this chapter. First we will elaborate on the multithreaded architecture of the framework and then we explain the functions of the most important blocks.

9.2.2 Framework Building Blocks

Multithreaded design

To make optimal use of the computer's hardware, VRment supports multiple threads running concurrently. Figure 9.1 shows a high-level view of the different threads. Centrally in the framework design, the 'Main Thread' coordinates the ensemble of all threads. It maintains a dedicated scenegraph that holds the master copy of the 3D scene. The structure of this scenegraph is mainly inspired on the work presented in [Raymaekers et al., 1999].

Depending on the version of the framework, the visualisation of the 3D world may be performed in a separate thread. This thread, supporting the OGRE engine runs independently from the application and maintains its own optimised scenegraph. The same is true for the collision engine keeping track of its own scene representation and allowing an application to calculate collisions

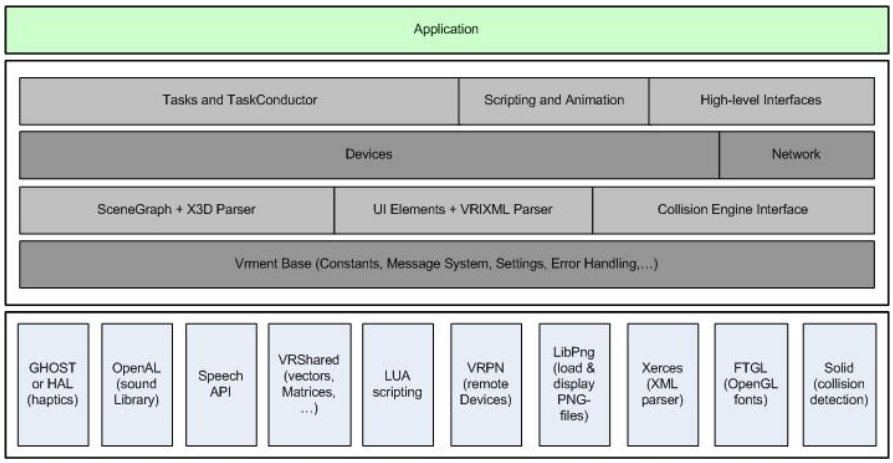


Figure 9.2: VRment Functional Blocks

asynchronously.

When using VRPN for device abstraction (as will be explained in section 10.2.3), the haptic scene is maintained at the VRPN server, and synchronised by the VRPN thread. Other versions of the framework, not using VRPN, keep the haptic scene in a dedicated scenegraph, driving the haptic device from a separate 'Haptic Thread' [De Boeck et al., 2000]. Finally, as will be elaborated in section 9.3, an optional network thread is responsible for distributing the main scenegraph across a TCP network.

The synchronisation and a thread-safe communication between all threads is maintained by means of a messaging system. A thread may connect to this messenger and subscribe for certain messages. Each subscriber then receives the relevant messages in a thread safe manner, coding for updates or queries. A detailed practical example of the messenger system is explained in section 9.3.

Functional Parts

As shown on figure 9.2, VRment can be seen as a ensemble of several functional blocks. At the very lower end, the most basic libraries can be found. These packages, mostly third party provide specific functionality that can be used throughout the entire framework, such as vectors, matrices, collision detection, XML parsing,...

In the bottom layer of the actual framework, we find the basic framework

functionality. This layer includes the 'messenger system', 'error handling' or 'Settings management'. The next layer above contains functions to hold the scenegraph (main scene representation) as well as functions to read scenes from X3D files. This layer is also able to read and visualise VRIXML menus and dialogs [Cuppens et al., 2004] and it contains the interface to the collision engine.

In the next layer, all functionality to communicate to external devices is grouped. This includes interfaces to the Speech API, PHANToM Device or audio feedback. We will discuss this layer further in section 10.2. The network is also designed in this layer; this will be explained in section 9.3. The topmost layer of the framework, is the 'application controller', and it collects functionality to manage the behaviour of the application, as will be explained in chapter 10. Here we find support for executing tasks, defining animations and support for scripting. Finally, on top of this framework the application specific code is situated, holding the programming code dedicated to the particular application.

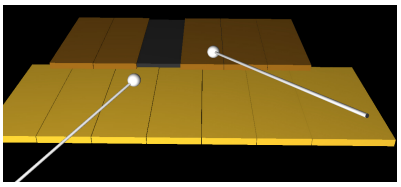
We will now explain how the messaging service is applied to support sending the main scene representation onto the network to support a distributed bimanual haptic simulation.

9.3 Support for Bimanual Haptic Interaction

The experiments with two PHANToMs providing two-handed haptic feedback, as presented in section 7.3 required an elaborated software design. As shown, earlier in section 8.3, adding a second haptic device to the same computer severely limits the complexity of the scene to be rendered. We therefore opted for a distributed design in which the haptic simulation is distributed across the network.

In this section, we elaborate on our multithreaded and distributed software architecture as we shortly explained in the previous section. The application which is used to prove the approach is called 'The Virtual Percussionist' [De Boeck et al., 2002b]. As shown in figure 9.3, it allows a musician to play vibraphone in a two-handed setup using force feedback on both hands. In a first section, we discuss some related work which founded the basis of our distributed solution. Next, we describe the multithreaded framework in more detail. In section 9.3.4 we validate our implementation by measuring the delay

(a) Virtual Percussionist Setup



(b) Virtual Percussionist Screenshot

Figure 9.3: The Virtual Percussionist

between the different modalities. Finally, we show how the implementation can be extended to support more dynamic scenes.

9.3.1 Related Work

For our distributed setup, we adopted some principles which can be found in telepresence applications [Reinhart et al., 2000][Preusche et al., 2005]. In those applications, typically one haptic device is present. This device may be connected in a master-slave setup, where one computer holds the virtual scene and another is responsible for the haptic simulation. VRPN [Seeger et al., 1997] is software that can be used to support such a remote access to several input devices. A recent extension designed in our lab and explained in section 10.2.3 adds support for holding a haptic scenegraph at the server side [Tailor II, 2006]. Other applications focus on the collaborative aspect of haptic virtual environments. These applications also support some kind of distribution of the simulation across the network, but the different simulations are operated by different users. Examples can be found in [Basdogan et al., 1998] [Basdogan et al., 2000], [Hespanha et al., 2002] and [Alhalabi and Horiguchi, 2001]. Evidently, if networked virtual environments are a basis to support highly demanding haptic simulations, network issues are another problem to overcome. Wilson et al. [Wilson et al., 1999], propose some algorithms that support efficient network-based haptics. Matsumotoy et al. [Matsumotoy et al., 2000] focuses on the influence of network issues on the haptic simulation.

For our setup, we proposes a master-slave setup, where the master computer drives the simulation and one haptic device, while keeping a slave computer

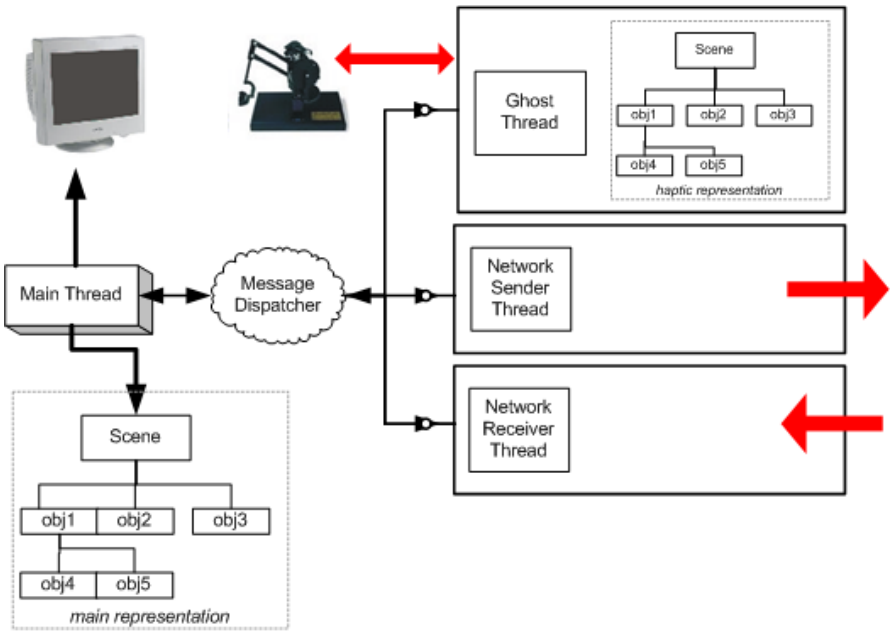


Figure 9.4: Thread scheme of the framework

synchronised. This slave computer drives another haptic device. For this setup, we suppose a reliable 100Mb ethernet network.

9.3.2 Framework Details

Multithread-communication

As already discussed in section 9.2.2, the VRment framework consists of a main scene representation and several peripheral threads that hold their own optimised scene representation. In the particular implementation of the 'Virtual Percussionist', we do not use VRPN to connect the PHANToM device, but drive the device directly from within the application as shown in figure 9.4. We also distinguish two network threads, one to receive data, and one to send.

The communication between the threads is maintained by a *message dispatcher*. When a thread registers to this system, a message queue is created. Messages sent to the dispatcher will be posted to all queues and arrive at their destination in thread-safe, first-in-first-out manner. This method provides us

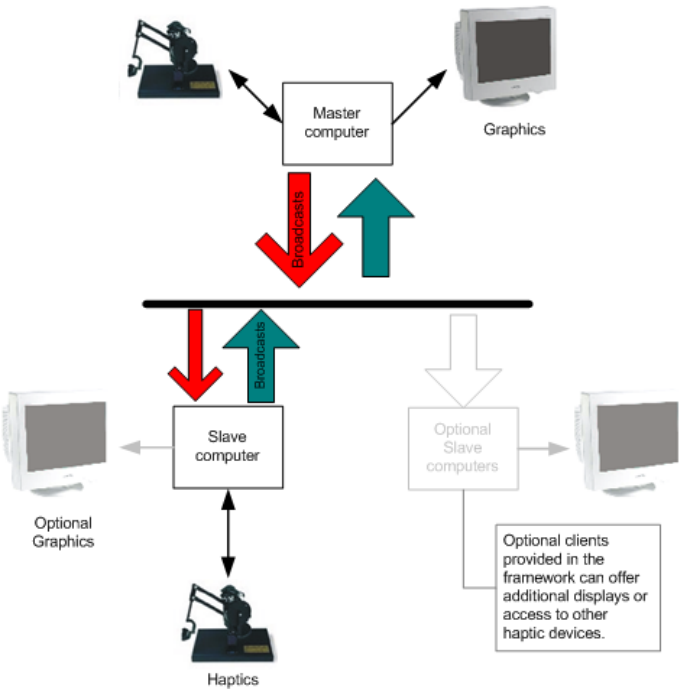


Figure 9.5: Network Scheme

with a very flexible method, with low overhead, to plug in and out new functionalities (such as network support, haptics, collision detection, etc) without redesigning the software architecture.

When objects in the main scene change, messages are posted, updating all peripheral representations such as the haptics scene and the network.

Network Communication

Seeger et al. [Seeger et al., 1997] and Reinhart et al. [Reinhart et al., 2000], propose a network solution optimised for performance, in which graphics and haptics are distributed among two computers: the haptic computer runs as a slave from the graphical workstation running the master copy of the simulation. We adopt this approach in our design, although our aim is to support *more than one* haptic device. Therefore we also look at the collaborative haptic environments, which often share a similar architecture in which the master computer supports haptics for the first user, while the other PHANToMs are

connected to slave computers [Alhalabi and Horiguchi, 2001]. However, some applications use a dedicated server only holding the master scene [Matsumotoy et al., 2000]. Although we prefer the first setup, in which the master computer drives one haptic device and the slave the other, Matsumotoy's approach can be implemented as well, with little extra coding.

As shown in figure 9.5, master and slave computer communicate via UDP broadcasts. UDP has the benefits over TCP to cause less network overhead, but it requires a reliable network. Using broadcasts or multicasts instead of point-to-point communication, allows us to easily extend our design to even more haptic slaves, or to add two or three display slaves to support multidirectional views. The master loads the scene from disk and broadcasts all changes over a particular port to the slaves. Later, the master uses the same port to transmit its camera and pointer-position, 25 times per second. The slave updates its internal information according to the information received from the master. At its turn, the slave sends the position and rotation of the PHANToM onto the network over another UDP port. The other listeners, in our case only the master, use this information to update their local information.

Figure 9.5 indicates that it is fairly easy to create more network nodes to realize other distributed services with just a little effort. Since every network node broadcasts its own information over its own UDP port, each additional network node just has to listen to the UDP ports of interest. If, for instance an additional slave for graphics should be desired, this slave computer just has to listen to the relevant TCP ports to receive all the relevant information.

In the next section, we will explain how the internal messaging system will continue on this modular flexibility in a multithreaded design.

9.3.3 Framework for the 'Virtual Percussionist'

The 'Virtual Percussionist' is the application we use to validate our approach. The application consists of a master and a slave part that both support a PHANToM Device that is used to play a virtual vibraphone keyboard. Both master and slave communicate over UDP as described above. The next paragraphs explain the specific implementation, based on the 'VRment' framework, in order to expose to the user a solid and coherent haptic experience when playing with two hands.

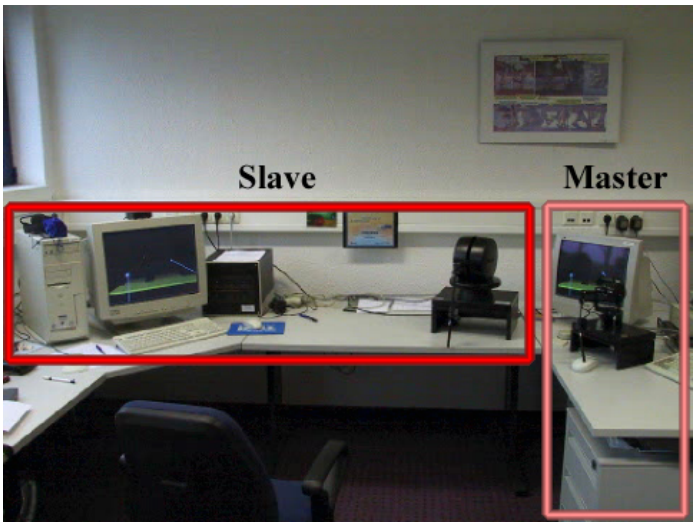


Figure 9.6: Master-Slave Setup in our lab

Master application

The master application loads and maintains the virtual scene, which in this case is a musical keyboard of one octave. While loading the scene from disk, every new object causes a message to be sent through the dispatcher. The master application contains two threads that have been subscribed to the message dispatcher listening for messages coming from the main thread (figure 9.7): the haptic thread, which drives the PHANToM for one hand, and the network thread, broadcasting the newly created scene onto the network using port 7777 (bold solid arrows in figure 9.7). After the scene has been loaded, we assume any slave computer to be up-to-date. During each iteration of the application, the main thread sends the coordinates of the virtual camera and the virtual pointers to the message dispatcher (bold dashed arrows). Only the Network Sender listens to those messages and broadcasts them over the network.

Slave Application

As can be seen from figure 9.8, the slave's application and thread design is quite similar to the master's, allowing us to reuse the majority of the code written for the master. The slave computer (possibly more than one) checks the network for the master's messages. The master can send two different types

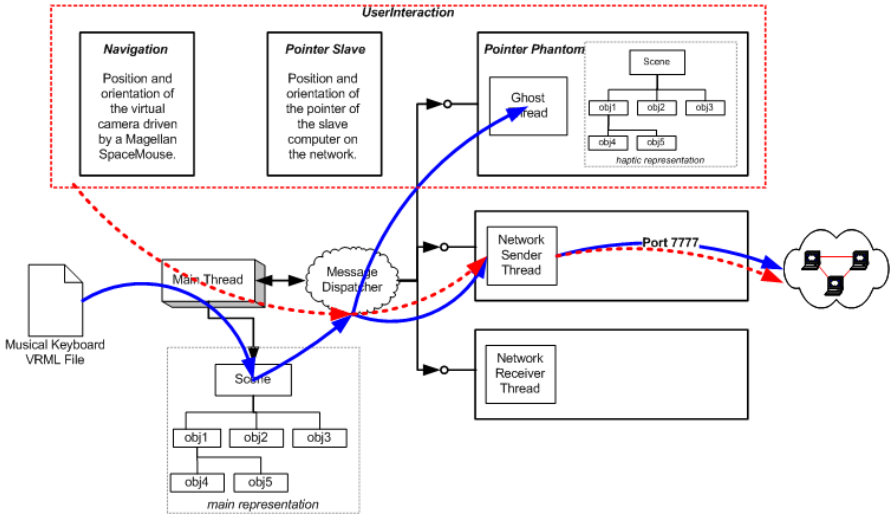


Figure 9.7: Message flow in the master application

of messages: messages containing scene changes (solid arrows) and messages containing a recurring update of the master's camera and pointer positions (dashed arrows coming from the network). Messages updating the scene are propagated to the GHOST scene (for haptic display) and to the main representation (for an optional graphic display). On the other hand, the main thread receives all messages of the master's camera position and virtual pointers. The position and orientation of the slave's PHANToM, will be sent back onto the network as can be seen by the short-long-dashed line in figure 9.8.

Discussion

As the slave instantly receives any scene updates from the master, the slave's haptic scene is consistent to the master's representation, causing both PHANToMs to display the same data, so we can assume (and of course we can feel) we have a consistent haptic sensation for both hands.

As the application is called 'The Virtual Percussionist', the application is supposed to generate music when the user strikes the virtual notes. All audio caused by contact of both PHANToMs is generated at the master computer: the master computer detects haptic contact of the attached PHANToM and directly sends it to the main thread to produce the audio. The slave computer also detects haptic contact (because it is the same code as the master's) and

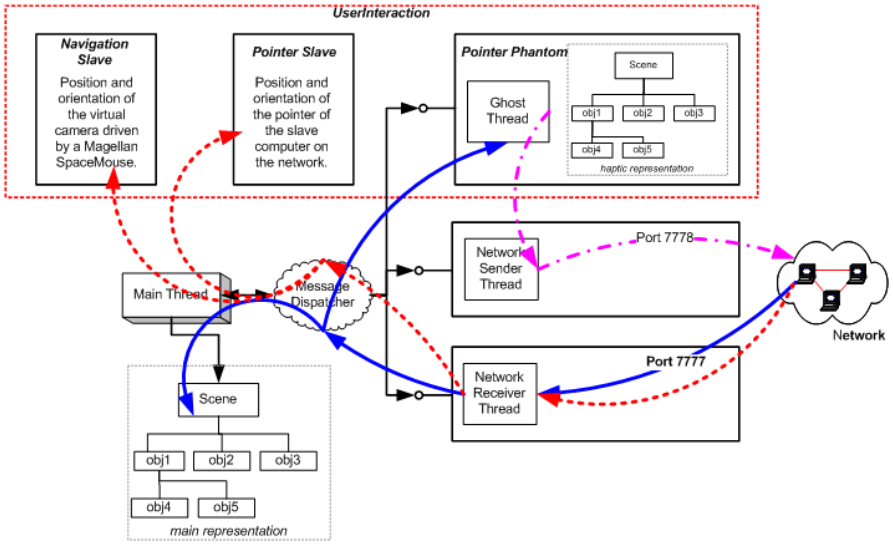


Figure 9.8: Message flow in the slave application

sends it to its main thread. This main thread uses the information to prepare a specific message that will be sent over the network to the master computer. The master receives this message and generates the appropriate audio.

As can be understood from the previous sections, the above-mentioned concept leaves programmers with enough flexibility to adjust this concept to their own needs. Since the slave computer holds a copy of the entire scene and it also knows the position of its own PHANToM as well as the positions of the master's camera and PHANToM, the graphical output can also be provided by the slave (instead of the master). Also, an application on a third computer can be written to listens to the relevant ports. This additional computer can then drive a second display that is more to the left or the right, offering a broader viewing angle such as in the PSD. If some applications require another additional PHANToM (sometimes vibraphone is played with 4 mallets), another slave can be created with the same code as the current slave, sending its data over another UDP-port. In that case only the workstation outputting the graphics, must contain additional network-receivers to receive all pointer positions.

9.3.4 Assessment

In the previous section, we have described a network communication principle and the accompanying internal design to achieve a two-handed haptic sensation with a distributed haptic load. To assess the presented solution, we will evaluate the 'Virtual Percussionist' application to see whether or not the different output modalities (visual, touch and audio) arrive with a noticeable delay.

In the current application configuration, the right-hand PHANToM runs on the computer that outputs audio and graphics. Therefore, we can assume that there is no perceptible delay between those three modalities. Indeed, the graphic loop and haptic thread both run independently, reading from their own scene representation. The callback-function for haptic contact is the only synchronisation between both threads, so we can assume this delay is negligible.

The slave computer, on the other hand, has to send its PHANToM position (for graphical feedback) and its haptic contacts (for audio feedback) to the master. Those events have to travel down the messaging system of the slave and subsequently are transmitted over the network. When arrived at the master, the events have to travel up the master's messaging system again. It should not surprise that those operations can take a considerable amount of time.

When using the application, at first sight, there appears to be no noticeable delay between the three sensory feedbacks. Though, it is desirable to measure the delay between the haptics at the slave computer and the other output modalities at the master computer. Both modalities run on different computers, which makes measuring the delay in software difficult to establish. Alternatively, we have adopted the principles of [He et al., 2000]. Using a digital video camera we recorded the PHANToM haptic device, the video output on the monitor, and the audio produced by a speaker. The 'contact' of the haptic device with an object is approximated by one of the captured video frames.

In figure 9.9 subsequent positions (with an interval of 1/25th of a second) of the slave's PHANToM can be seen. The light shaded part of the picture indicates the bottommost position of the PHANToM. The third frame from the left can be considered as a good approximation of the contact point, as this is the bottommost position we can distinguish from the video frames. The audio stream indicates the audio output at the master about one frame

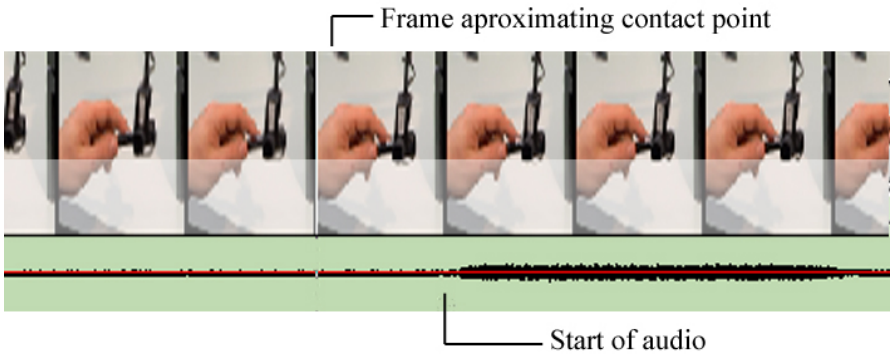


Figure 9.9: Frame sequence and audio when colliding virtual vibraphone

later. Therefore we can conclude that the delay between the haptic sensation caused by the slave and the auditory output is less than 1/25th of a second, or 40ms, which we consider as acceptable. It is generally accepted that a delay of up to two frames of the audio with respect to the video is imperceptible. Alternatively, a delay of one frame for the video in respect to the audio can be noticed.

Since haptic contact messages and messages updating the slave's PHANToM position are both sent in the same main thread's pass, we can assume they arrive almost together at the master. There they are processed in the same pass; therefore, we can assume there is very little or no delay between graphics and audio. With the same video technique as described above, we could confirm this assumption.

9.3.5 Extending the framework

The previous section presented a flexible framework which allows to distribute bimanual haptic feedback across a network. The assessment of the implementation showed minimal latency between the different modalities. Although it appears to be a valid approach to support more complex scenes as is possible with two PHANToMs driven by the same computer (section 8.3 and [De Boeck et al., 2002a]), this solution only supports static scenes in which each object's velocity is fixed at 'zero'.

This section describes an extension to this distributed bimanual haptic setup, introducing physical simulation of the objects, as well as interaction between

the two PHANToMs ([De Boeck et al., 2003c]). Physical simulation allows the user to push against an object with one PHANToM, giving it a certain velocity. While the PHANToM at the other hand can catch and hold the moving object. Another possibility is to push with both hands simultaneously against the same object in opposite directions, to feel the forces of one hand at the other. Future addition of friction can help to lift the object with both hands [Melder and Harwin, 2005].

Physical simulation at the master

To realise physical behaviour of virtual objects, the basic laws of Newton can be applied, although numeric approximation is not always a trivial issue. A lot of work already has been done here. In this work, the main calculations are based on the calculations that can be found in the work of Baraff et al: [Baraff, 1989] [Baraff, 1994] [Baraff, 1997]. As can be seen in figure 9.10, the physical calculations at the master application run in two separate integrators, each updating its own scene representations.

Low speed integrator

As a scene can grow complex, containing several moving objects, physical calculations can take quite some processing time. Hence, a full integration cannot be performed within the haptic loop. We thus have chosen to execute the main physical calculations in the (low-resolution) thread of the master computer ('Main Integrator' in figure 9.10).

In the main thread, when the master PHANToM touches an object, the feedback force is captured and passed to the integrator. Since the forces exerted by the user cannot be predicted or pre-calculated, they are sampled each 25th of a second and are considered to be constant over the interval. This means that the calculations can be done by simple Euler integration.

The main thread asserts that the integrator handles all objects that undergo a force.

The force $\overrightarrow{F}$, implies an acceleration

$$\vec{a} = \frac{\vec{F}}{m} \qquad (Newton's Law) \tag{9.1}$$

If we know the time interval since the last integration, we can calculate the velocity

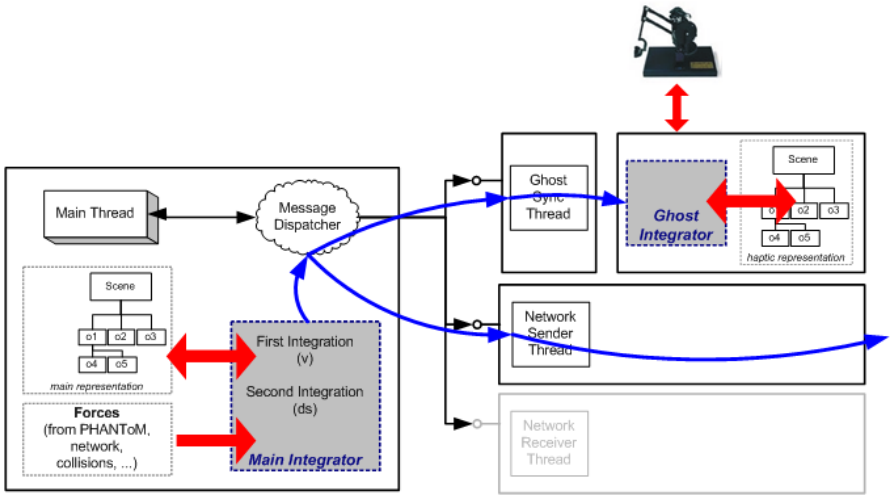


Figure 9.10: Physical Integration at the master

$$\vec{\Delta v} = \vec{a} \cdot \Delta T \qquad (First \;\; integration) \tag{9.2}$$

and next the displacement

$$\vec{\Delta s} = \vec{v} \cdot \Delta T \qquad (Second \;\; integration) \tag{9.3}$$

The integrator finally updates the position of the integrated objects. After the first integration, all objects that have a velocity greater than zero are kept in a list so that they automatically will continue their second integration in the next integration loops.

Synchronisation with the high-speed haptics

When the main integrator has finished its work, a list of all integrated objects, together with their velocities and positions, is sent through the message dispatcher. The 'Ghost Sync Thread' receives the message and interprets the values to update the haptic representation. In a first step, the velocity calculated at the master's integrator is copied into the haptic object.

Since the GHOST integration is done at another update rate (1 kHz) than the main integration, changing velocities will certainly originate a mismatch between the position calculated by the master integrator and position of the

haptic object. Therefore, we calculate a correction velocity, a velocity with a limited life-time (empirically set at 80 ms, thus two main thread-loops). This correction velocity brings the object smoothly, within the given time interval, from its current position to the position dictated by the master integrator. Each haptic loop, the GHOST thread adds the correction velocity to the given velocity and integrates the result to a new displacement, which moves the haptic representation of the object.

Physical simulation at the slave

Up until this point, only physical behaviour at the master computer has been described. Figure 9.11 depicts the high level overview of the calculations at the slave computer.

Synchronisation on the network

When the master computer's main thread has sent the results of the integrator to the message dispatcher, the 'network sender thread' also receives the packet. The framework now is able to deliver the data to the slave computer (dotted arrow in figure 9.11).

When the client has processed the message, all data is taken up by the main representation of the client. Next, the second integration, which basically is exactly the same as the code of the master, is performed. Although positions and velocities are directly updated from the master, executing this integration code at the slave diminishes the effect of network latencies.

After the slave's integrator has done its work, positions and velocities are sent to its haptic representation to perform similar haptic calculations as those on the master computer (solid arrow in figure 9.11).

Sending back the forces to the master

Since the main integrator at the master computer is the master for the entire simulation, the slave does not execute the first integration step. This means that, when the PHANToM at the slave computer interacts with an object, the executed force must be sent to the master computer. At the master computer the force will be handled as if it is a local force. This allows us to push two objects simultaneously, or push the same objects from opposite sides, causing the forces to neutralise.

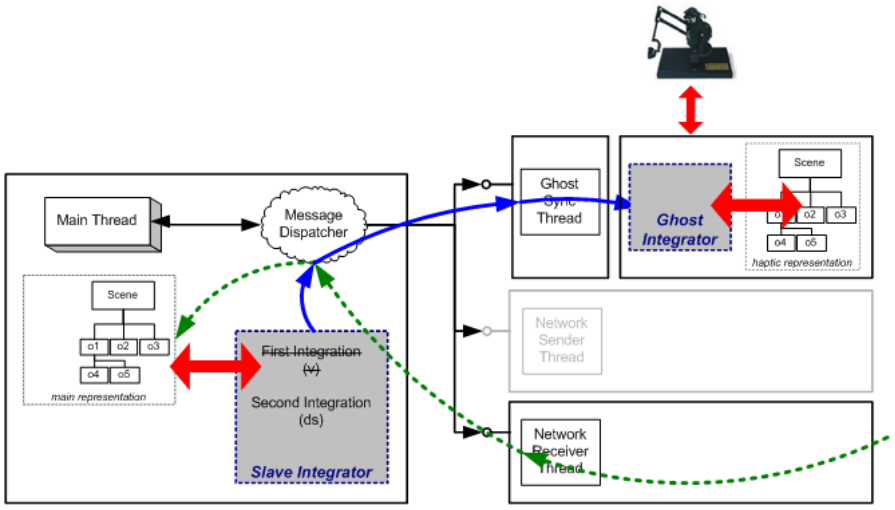


Figure 9.11: Physical Integration at the slave

Results

In the previous sections, we have already shown that the distributed simulation results in a stable and solid haptic experience not suffering from network or messaging delays. Testing the new extensions for a more physical behaviour, we can see that a user is able to push, throw and catch objects. Additional gravity to each object even gives an extra sense of realism.

To evaluate the stability of our simulation, we have logged the magnitudes of the correction velocity, calculated to correct a possible drift of the haptic integration. At the master computer, values lay between 0.002 and 0.14 by a maximum real velocity of 1.09. This results in an average proportion between correction- and real velocity of 7.0% (median value 2.0%). At the slave, the correction velocity lies significantly higher: averagely 25% of the real velocity, but peak values seam to disappear within a couple of graphic loop frames. If we look at the direction of the correction velocity we see that this velocity corrects the delay of the haptics loop: when accelerating, the correction velocity is in the same direction, when decelerating, the correction velocity is opposite. Since the correction velocity is calculated to correct the error during an interval of 80 haptic frames, but is updated each 40 haptic frames and since the correction velocity can change discontinuously, no overshoot occurs due to this correction.

9.3.6 Applicability

Although the framework for distributed bimanual haptic interaction establishes an experience, which works well, we are still confronted with the following two problems. First, as the velocity of a moving object rises above a certain value, the PHANToM sometimes pops through the object when trying to bring the object to halt. We expect that this problem may be solved by rewriting the collision detection function of our GHOST objects. A second problem occurs at a single processor machine: as soon as the haptic integrator starts its calculations, at random intervals, the haptic thread blocks the entire machine (inclusive mouse movements) for a couple of seconds. Even the GHOST's time interval guard doesn't notice the interruption. Strangely enough, if we run the application on a dual CPU, the average processor load stays below 20%.

Finally, even more important, although this solution provides haptic feedback at both hands, the lack of a proprioceptive feeling between the dominant hand with respect to the non-dominant hand, makes this approach less suitable to fit in our solution, as we already mentioned in section 7.3. However, we are sure that the proposed approach of spreading the haptic computation across the network, in general may be valuable in other practical setups.

9.4 Summary

In this chapter we elaborated on the software design, which forms the basis to support the research presented in this thesis. In a first section we described the main building blocks of 'VRment', our in-house research framework. In section 9.3, we explained how the messaging system may be applied to distribute the haptic scene across the network, providing the user with a solid two-handed haptic experience. Although, technically spoken the implementation works well, from a usability point of view (as explained in section 7.3) this approach seems to be less suitable to fit in our overall solution to improve the interaction in 3D environment, as it lacks the feeling of proprioception.

Chapter 10

Supporting Multimodal Interaction

Contents

10.1 Introduction

In this chapter, we elaborate on our efforts to simplify the design process of a multimodal interface, so that a designer can easily create, evaluate and modify a multimodal interface, for instance by changing input devices or adapting interaction techniques. In a first section, we explain how we designed the

'Devices'-layer in VRment (figure 9.2) to make abstraction from the implementation details of each device. In section 10.3, we show how this design can be further extended to support the automatic execution of NiMMiT diagrams.

10.2 Multimodal Input

10.2.1 Modalities and Devices

In a multimodal application, input and output are provided by using different devices. Due to their specific characteristics, each device or modality communicates in its own manner to the application. This means that a lot of coding and recoding is necessary for each concrete application. In this section we will describe some frequently used modalities and their properties to an application. We concentrate on three modalities in particular.

The first modality is direct manipulation (with force feedback), which is suitable for direct interaction within the 3D world [Stone, 2000]. In our case, this functionality is provided by using a PHANToM device. The PHANToM requires the application to poll for the current pointer position. On the other hand, the event of 'touching an object' requires a specific (custom) call-back function for the communication to the main application.

Secondly we consider interaction through widget manipulation [Raymaekers and Coninx, 2001], such as menus and dialogs (see figure 10.6). The widgets are 2D objects positioned in 3D space and are accessed by the PHANToM, as well. Because of their nature we call them 'haptic hybrid 2D/3D widgets'. We can consider the PHANToM to have a two-fold purpose: it can either be used for direct manipulation, with a direct result in the virtual world, or it can be used to activate widgets, which results in a command that is activated. It is clear that both interactions cannot be used at the same time.

A third modality is established by a speech interface [Cohen, 1992]. The actions provided by this interface are defined in a grammar-file coding for the phrases that can be recognised. When recognition takes place, the API requires another custom call-back function to be activated.

We have chosen for the modalities listed above, as they directly apply to the experiments in part II of this thesis. Other applications may require other modalities (e.g. gestures) that have to be implemented in a different manner. We believe, however, that with the proposed combination of modalities we can

handle most situations. Other devices will often be implemented as a second pointer [De Boeck et al., 2004b] or will also use a call-back mechanism.

10.2.2 An Event Driven System

Based upon the mechanisms described in the previous section, we now propose an event model that abstracts from the particular device properties. This section will cover the programmatical aspects of our solution. In section 10.3.3, this solution is part of a practical example.

Creating events

Figure 10.1 shows the class diagram getting data from the devices. The shaded classes can be considered as adapter classes as their aim is to convert the individual interfaces of all input-channels to a common interface that is usable by the application core. Some modalities, such as speech recognition, manage their own thread and provide their results by a call-back mechanism, implemented in the 'speech'-adapter class. Other modalities, such as direct manipulation, receive a time-slice to execute code for the abstraction (driven by the class 'VirtualPointers'). Please note that the abstraction only concerns the input to the system; the synchronisation of the haptic scene is maintained by the messaging system as described in section 9.3.2. Finally, 'UIContainer' is a class that is responsible for the interaction with widgets such as menus and dialogs. This class groups the instances that initialise from VRIXML [Cuppens et al., 2004](section 7.4.1) and manages the menus and dialogs. When the user interacts with a widget, a call-back-function is executed.

As mentioned before, the PHANToM is used for both direct manipulation and widget interaction. This means that either the direct manipulation in the virtual world, or the action within the widget has to be executed. Never both of them. This problem will be solved by the *InteractionLayer*, which asks the adapter classes if one among them wants to process the current pointer-position. If one does so, this event is performed and no direct manipulation event takes place. Likewise, the application can indicate that it wants to receive all pointer information without it being processed by any other adapter class. For instance, if the user gives continuous input to the program (e.g. to move an object), the application can avoid involuntary interaction with a menu.

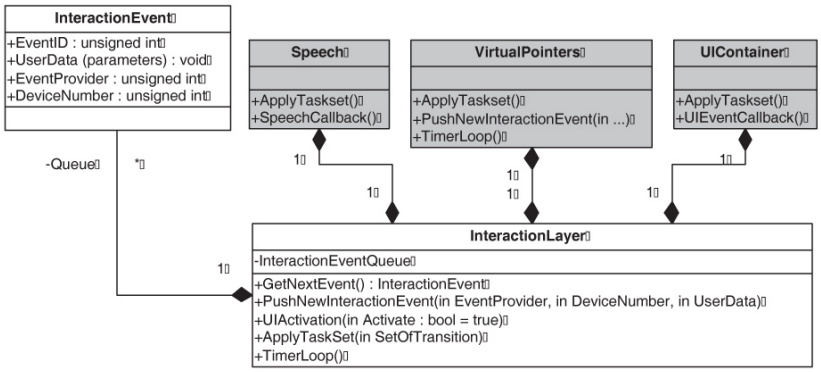


Figure 10.1: Abstraction of input channels

We can conclude that all adapter classes ultimately execute a given function or a call-back function when an event occurs. Depending on the nature of the modality, the event can be a button-click, a recognition of a spoken command or a dialog-event. The aforementioned function now creates an event record by specifying the necessary parameters and posting it in an *EventQueue* in the *InteractionLayer*.

Handling Events

All events coming from the input channels are collected in a queue by the *InteractionLayer*. Each event record contains a unique event identifier and an array of parameters. The core of this abstraction is the *TaskConductor* (figure 10.2). This class is responsible for reading the events and executing them. We define an abstract class 'Task'[1], which contains the common functions to perform a specified operation. The developer has to derive this class and implement the application specific logic for each task, but we will cover this in more detail when we elaborate on the automatic execution of NiMMiT diagrams in Section 10.3.4.

A problem arises when we want to apply this principle to 'direct manipulation'. Indeed, it may be clear that direct manipulation cannot be abstracted to discrete events that fire a task. Instead, the input is continuous. To tackle this

[1]The name 'Task' has been chosen in a historical context which was not directly related to model-based development. In the context of model-based design, this name may be confusing because a 'task', as defined in a ConcurTaskTree, not necessarily corresponds with a 'task' defined in this design.

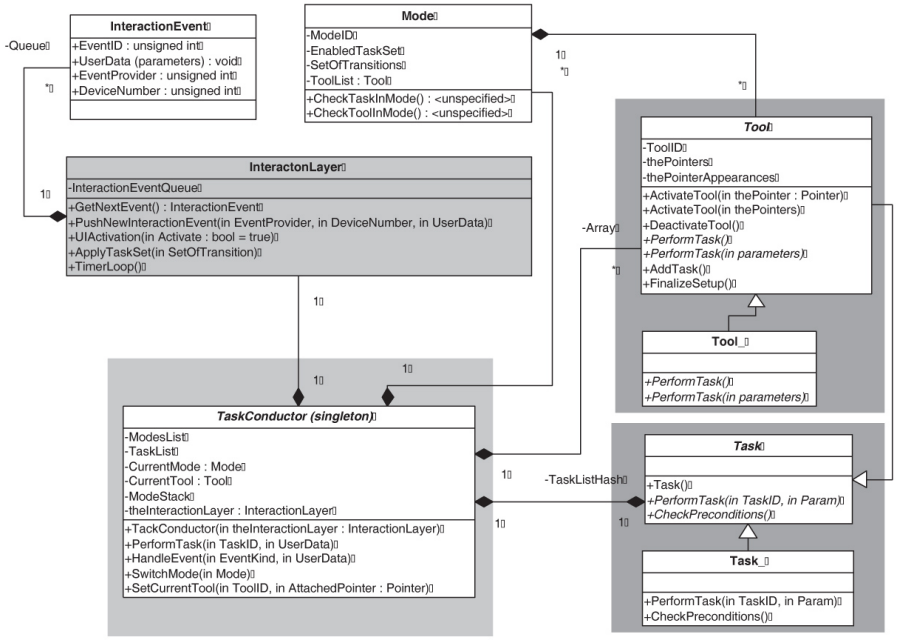


Figure 10.2: Event handling

problem at this time, we define the concept of a 'tool'. Later in section 10.3.4, however, we will explain how we can improve our design by using NiMMiT diagrams instead. In fact, as the class *Tool* is derived from *Task* (as can be seen from figure 10.2), a tool can be seen as a continuous task, and hence it also contains application specific code. A soon as a tool becomes active, it constantly receives the events of the devices for direct manipulation.

10.2.3 Device Abstraction Via VRPN

The aim of the abstraction above is to provide VE-developers with a framework by which they can easily create multimodal applications. With our approach, an abstraction is established interfacing the different characteristics of the devices and modalities, for as long as the adapter classes already exist.

However, when an application uses different devices for direct manipulation simultaneously, or when the developer wants to experiment between several alternative devices, this abstraction is not sufficient, since it still requires a lot of coding for the actual implementation. Therefore, we decided to adopt

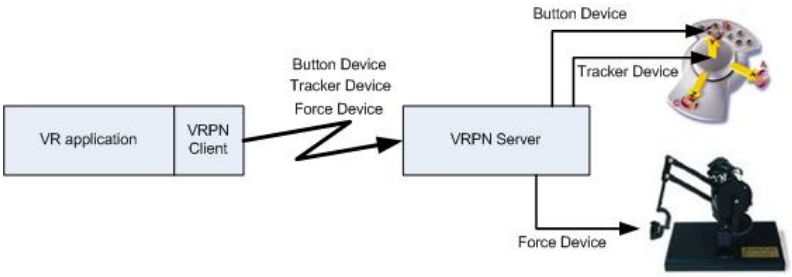


Figure 10.3: VRPN principle

VRPN (Virtual Reality Peripheral Network) [Tailor II, 2006][Taylor II et al., 2001]. VRPN makes abstraction of the concrete input device, by classifying them in a limited set of categories: tracker, button device, haptic device, analog inputs and sound. The actual devices are hosted by a dedicated PC, running the server software. An application that wants to get input from a device, implements the client software and connects to the host PC for the desired *abstract* device as shown in figure 10.3

Although VRPN has proven its benefits in many applications around the world, its support for haptic feedback is rather limited and very experimental. In order to make VRPN suitable in our framework, we therefore decided to extend VRPN with a scenegraph for the PHANToM. We are proud that this extension has been formalised in version 7.0 of VRPN.

Figure 10.4 shows the class diagram of the extension. The left hand side of the schema depicts the classes that are extended at the server. At the right hand side, the classes at the client are shown. Generally spoken, all relevant scenegraph manipulations that may be performed by the client are interfaced in the class '*ForceDeviceRemote*'. Those interface functions send a specified message, using the VRPN network system. When arrived at the server, the message is unpacked, interpreted and applied to the local scenegraph, kept in the '*PHANToM*'-class.

The extension works well, and it is still used as the standard setup of our framework, rather than connecting the PHANToM directly to the PC. Two limitation, however, are currently known. The extension only supports static scenes, which means that no moving or deformable objects are supported. For a possible implementation of moving objects, we refer to our solution explained in section 9.3.5; deformable objects may be implemented as shown in [Raymaekers et al., 2005b]. A second limitation of our extension is that

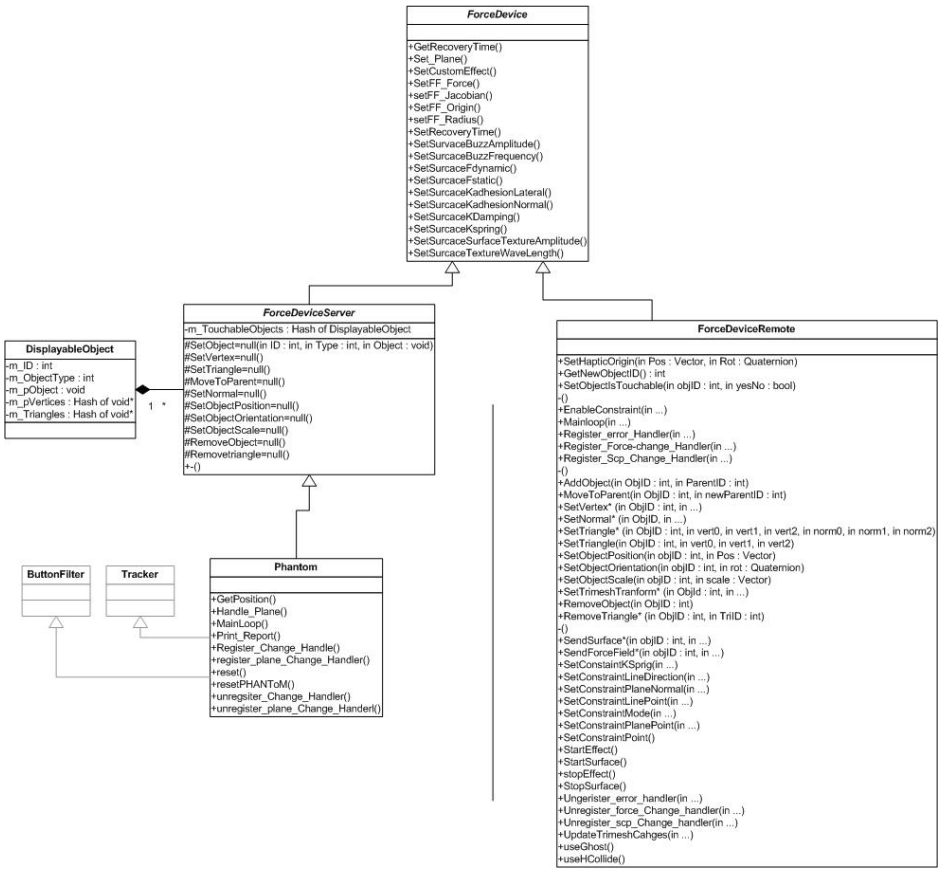


Figure 10.4: VRPN Extension Class Diagram

it requires the GHOST API and the PHANToM device to be used. For our particular research, this was not a limitation since in our lab, we only use a PHANToM device for force feedback. If other haptic devices are desired, the design can be reused, but it may be implemented using another API, supporting other devices.

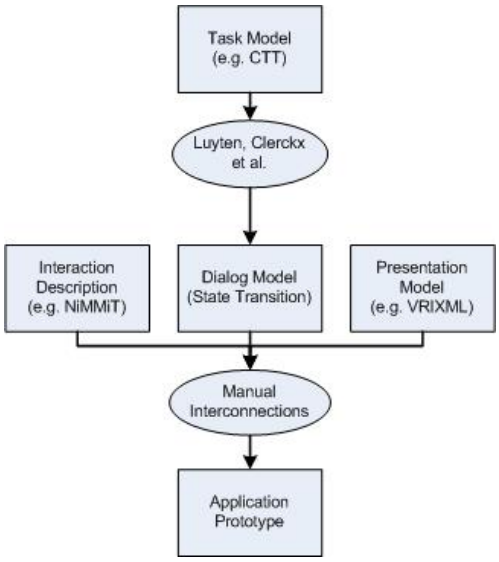


Figure 10.5: the VR-DeMo Process

10.3 Running NiMMiT Diagrams

10.3.1 NiMMiT as a Part of a Model-Based Development Approach

In the previous sections, we explained how the VRment framework has been designed in order to make abstraction from the specific implementation of the device and the modality, and how this abstraction can be optimised using VRPN. Still, a lot of programming is necessary for the application specific code such as the implementation of the interaction techniques or the enabling and disabling of certain actions. The VR-DeMo project (IWT 030248) [EDM-WISE, 2004], from which this thesis in some degree is part of, investigates how the model-based development approach, known from other domains such as mobile and pen-based devices, may be applied to virtual environments.

Figure 10.5 shows the basic process proposed in the VR-DeMo project. Starting from a task analysis, a designer may start with a high-level description, such as a ConcurTaskTree (CTT) [Paternò, 2000]. Based upon existing algorithms, such as the algorithm proposed by Luyten et al. the tasks with their temporal relations in a CTT can be converted to a list of 'enabled task-sets' [Luyten, 2004]. An 'enabled task-set' (ETS) defines which tasks may

be enabled at the same time. The result of this conversion, the ETS and its tranisitions, is called the dialog model[2]. Alternatively, in our current implementation, this model with transitions between dialogs, may be seen as a state transition network [Parnas, 1969] with each state corresponding to an ETS.

Analogously, in the VR-DeMo approach, the dialog model also defines an ETS, the set of tasks that are available in this state of the application. However, since we are dealing with 3D multimodal interfaces, the dialog model does not necessarily have a direct relation to a 'dialog' (as we know it from WIMP interfaces), which is mostly the case in 2D interfaces. The 'enabled tasks' in a given state, may be triggered by means of user interface elements (UI elements), such as dialogs or menus, or by an elaborated interaction technique, such as 'grabbing an object'.

Further in the VR-DeMo approach, as seen in figure 10.5, the UI elements are described in the Presentation Model, using VRIXML [Cuppens et al., 2004]. And we propose to 'describe' the interaction technique using NiMMiT.

Finally, the Dialog Model, interconnected with the Presentation Model and the Interaction Model, are converted to an application prototype.

As it falls beyond the scope of this thesis to elaborate in detail on the open issues and benefits of this model-based approach in 3D multimodal environments, we refer the interested reader to [Cuppens and Coninx, 2005], [Cuppens et al., 2005] and [De Boeck et al., 2006b]. In the next section, we will shortly explain how the 'enabled task-set' is connected to the event system we defined in section 10.2, and this is illustrated with an example. Finally, in section 10.3.4, we describe how the NiMMiT notation (introduced in chapter 4) can be applied for automatic execution of the model.

10.3.2 Using the Event System

It may be clear from the previous section that the Dialog Model defines the set of tasks that may be enabled in a certain application state. The core of the event system, the 'TaskConductor' (figure 10.2), holds a list of all application states using the *Mode*-Class. A state contains a list of possible tasks ('enabled task-set') and a list of possible tools (on which we will elaborate in

[2]The dialog model describes the human-computer conversation. It specifies when the end user can invoke functions through various triggering mechanisms (push buttons, commands, etc.) and interaction media (voice input, touch screen, etc.), when the end user can select or specify inputs, and when the computer can query the end user and present information [Puerta, 1997].

section 10.3.4). When a state becomes active, the task conductor sends the 'enabled task-set' back to the 'InteractionLayer' so that the given menu and speech commands can be enabled or disabled.

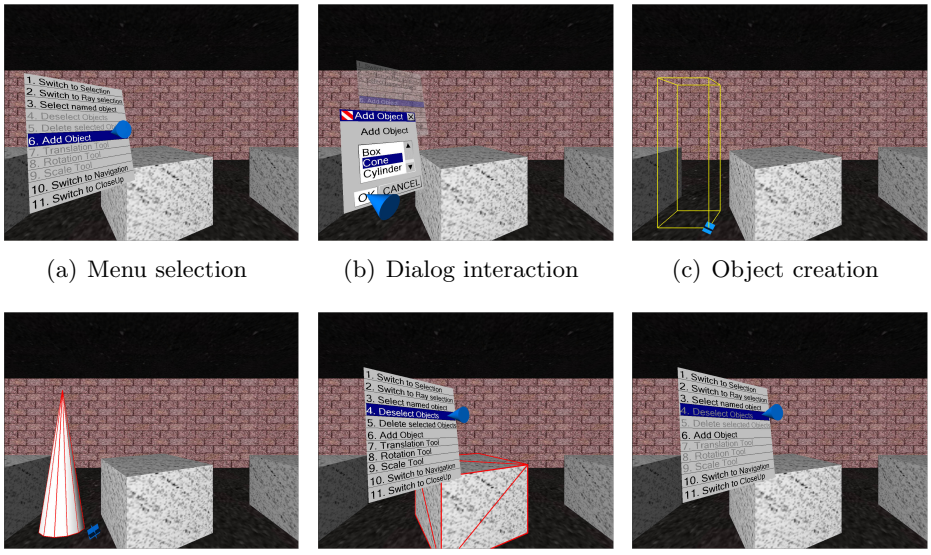
From the first experimental examples, however, it quickly became clear that the enabling of a task not only depends on the current mode ('enabled task-set'). For example, the *paste*-command may be available in all application states, but it has little sense to activate the command before an object has been copied. We have chosen to implement this extra condition, which is specific for the task itself, as a *precondition*, available in the 'task'-class. It may be stressed that 'preconditions' in this context are situated at a much lower level than the preconditions known at the CTT level. In our situation, a precondition may query information which is only available in the internal state of the application, at the implementation level. After each action, the task conductor now will ask each task in the 'enabled task-set' to verify its precondition. If its result is false, the task will be disabled from the current 'enabled task-set'. The final (modified) 'enabled task-set' is then passed to 'InteractionLayer', enabling the correct items.

10.3.3 Example: 3D Multimodal Modeling Tool

Upon the proposed scheme of events, states and 'enabled task-sets', a proof of concept application was built. This application fits within the scope of our research to enrich the haptic experience. The application presents a 3D multimodal modeling tool that allows the user to interact using direct manipulation with force feedback, speech commands and widget control. We have chosen this example, because it is closely related to experiments described in Part II of this thesis, and hence demonstrates the applicability of the aforementioned design directly to our research.

Menu commands

If the user shows the main menu of the application, several commands are available as shown in figure 10.6. If 'Add object' is chosen (10.6(a)), the main menu moves to the background and a sub-dialog is shown (10.6(b)). At the end of the menu interaction, the framework fires an event to the task conductor. In this case the event directs the framework to activate a tool (object creation tool), which, once activated, continuously receives the PHANToM position and orientation and manages the creation of the chosen object (10.6(c),10.6(d)).



(a) Menu selection (b) Dialog interaction (c) Object creation

(d) Created object (e) Selection preconditions (f) No selection

Figure 10.6: Creation of a new object using the widgets and checking preconditions

Likewise, commands such as enabling the 'selection mode' can be executed using the main menu. This command, in its turn, activates a task that switches application state. Next, a new task set, a new default tool and a new default widget set (e.g. a menu) are loaded and activated. In this example, all tasks that enforce a mode-transition can be activated using speech as well. In case of a spoken command, a similar event is fired which results in the execution of the same particular task.

Checking preconditions

As we have seen in section 10.3.2, the enabling and disabling of a task cannot be expressed in an 'enabled task-set' alone, therefore the idea of preconditions was introduced. Figure 10.6(e) shows the application in the main mode, with one object selected. The command 'Deselect object' is a member of the task-set of this particular mode and so it is expected to be enabled. After the user has executed the command 'deselect object', the 'enabled task-set' is again verified for its preconditions. In this case, the precondition 'is any object selected' is not satisfied anymore, and hence the command is disabled, as can

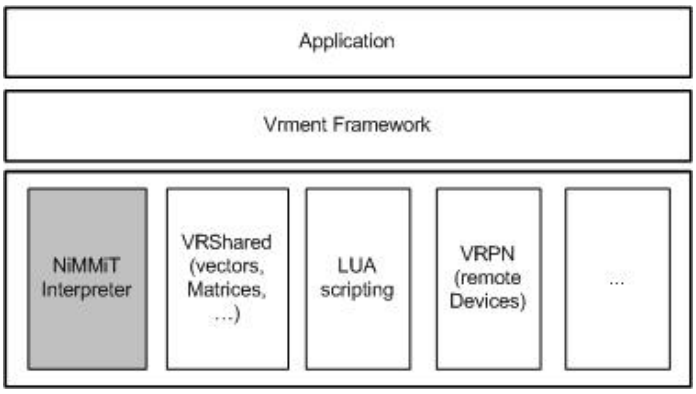


Figure 10.7: The NiMMiT interpreter as an external API

be seen from figure 10.6(f).

10.3.4 Interpreting NiMMiT Diagrams

Up until now, the handling of direct manipulation, such as creating an object 10.6(c), has been handled by the concept of a 'Tool', which is a class derived from the 'Task'-class, containing application specific (ad-hoc) code. Throughout this thesis, and in particular in chapter 4 and section 10.3.1, we have shown how NiMMiT may be suitable to describe multimodal interaction techniques. Responding to the continuous movements of devices for direct manipulation and listening to other events, coming from the speech engine or the dialogs, a NiMMiT diagram may now replace the concept of a 'Tool'.

The NiMMiT interpreter, is designed to be loosely coupled with the 'VRment' framework, which must allow us to apply the software in frameworks and domains other than 'VRment' and 3D user interfaces. Therefore, we can situate the interpreter next to the external APIs at the bottom frame of figure 10.7 (which is a part of figure 9.2).

Given the design of figure 10.2, the 'TaskConductor' checks if the current application state requires an interaction technique to be started. When necessary, it starts a 'NiMMiT Runner' and passes the incoming events to this interpreter. Loaded from the NiMMiT XML syntax, the interpreter holds an internal model as shown in figure 10.8.

The class diagram mainly reflects the structure as it is dictated by the diagram. In this way we can recognise at the class diagram that 'States' have one or

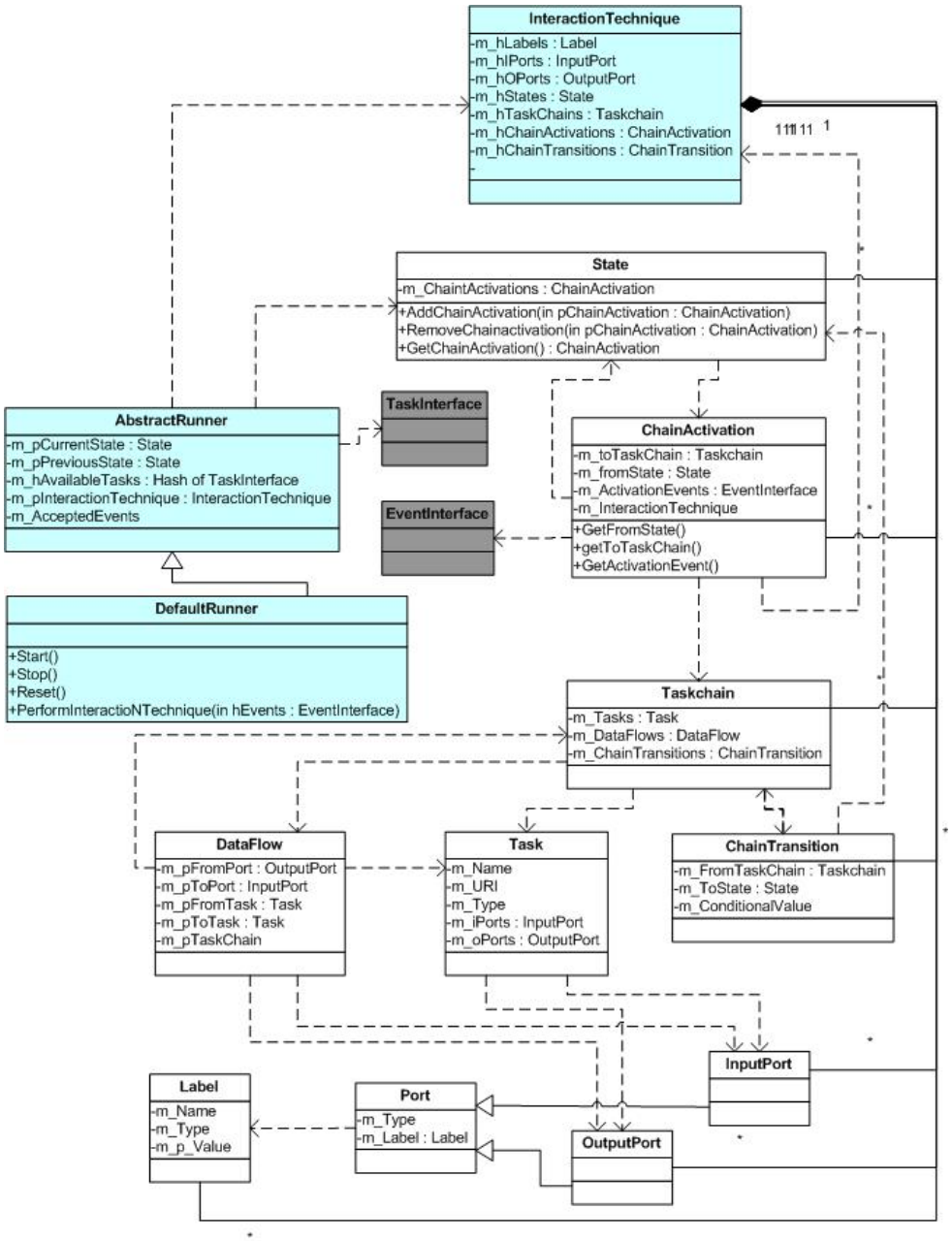


Figure 10.8: Simplified Class Diagram of the NiMMiT Interpreter

more outgoing arrows ('ChainActivations'); those 'ChainActivations' activate a 'TaskChain', containing 'Tasks' and a summary of the 'DataFlow', as well as one or more transitions to the next state ('ChainTransition'). 'Tasks' at their turn have input and output ports, which may be connected to a 'label'.

The 'DefaultRunner', derived from the 'AbstractRunner', conducts the execution, by keeping track of the current state of the current interaction technique. The 'InteractionTechnique'-class at its turn may be seen as a placeholder for all components of the diagram.

Based upon an appropriate combination of incoming events, the 'Runner' may execute a task chain, and hence a sequence of tasks. Those tasks (from the class 'Task') all have a unique name, which corresponds to the key-value of the 'm_hAvailableTasks'-member in the 'AbstractRunner'. Since we applied the 'TaskInterface' to all tasks, depicted in figure 10.2, we can automatically execute those 'VRment' tasks.

It may be clear that when generalising NiMMiT to other domains, the framework in which NiMMiT is incorporated, typically must provide 'standard tasks'. In our particular situation VRment comes with tasks such as moving or deleting objects, collision detection, etc. If functionality other than provided in the standard tasks is desired, such as for specific calculations, tasks can also be scripted using LUA-script. This is crucial for NiMMiT, since NiMMiT is not a programming language and hence does not support mathematical operations or the common programming structures.

Finally, to fully support multimodal events, incoming events are buffered in some sort of 'Melting Pot' as described in [Nigay and Coutaz, 1995]. NiMMiT supports the coincidence of multiple events 'at the same time', but we may assume that in terms of the execution-rate of a computer, 'simultaneous input' coming from the user is never exactly 'at the same time'. Therefore, each event which is not handled immediately, remains for about 1 second in the 'Melting Pot', after which it expires.

10.3.5 Support for Probes and Filters

As explained in section 4.6, NiMMiT can also be used to collect data that may be used for the statistic evaluation of the running interaction technique by using the primitives such as Probes, Filters and Listeners. The implementation of those primitives is quite straightforward as it results in the addition of some extra classes: 'Probes' directly capture the data from the running diagram,

according to the place they are connected to. Filters are seen as 'Meta-probes', and therefore the abstract class 'Filter' is derived from 'Probe'. Since filters may execute specific calculations (such as counting, timing, etc.), each filter contains its own specific code, sharing the same interface from the base class. New functionality can be easily added (and later reused) by implementing a new derived filter-class. The same is true for 'Listeners': as several listeners exist that output data to File, Screen or TCP network, they are all derived from an abstract class 'ProbeListener'. Finally, all probes and filters are coordinated by a specialised class 'ProbeDispatcher'.

10.4 Summary

In this chapter, we elaborated on the abstraction process to handle the implementation differences between different devices and different modalities. We have described the design of an event system, and how an extension of VRPN may support further abstraction. Thereafter, we shortly explained the model-based approach, as investigated in the VR-DeMo project in order to facilitate the development of multimodal 3D environments. With relevance to this thesis, we discussed how the generated events and the 'Dialog Model' can cooperate. Finally we presented the design of a NiMMiT interpreter allowing NiMMiT diagrams (discussed in chapter 4) to be automatically executed.

Chapter 11

Conclusions

11.1 Summary

Referring to the title of this thesis, the focus of this research was two-fold: from both a user's and a designer's perspective, we investigated how the interaction with a 3D environment can be improved, making optimal use of the human's multimodal communication capabilities.

In a first part of this thesis, we started by defining the most important definitions that are used throughout this dissertation, as well as some existing theories on multimodal interaction. We elaborated on the dialog between the human and the computer, and showed how 'metaphors' can be used to transfer the user's knowledge already known from another domain, to the new situation. Next we showed some well known interaction metaphors, which are commonly used in 3D environments. Finally, we proposed NiMMiT, Notation for MultiModal Interaction Techniques, as a solution to easily 'describe' multimodal interaction techniques.

The second part of this thesis, focuses on the usability aspects of the multimodal interaction. We proposed the 'camera in hand' metaphor, a solution to move the 3D camera position using force feedback. This solution appeared to be beneficial especially for novice users, who do not have any experience in 3D environments. An improvement, in order to address the remarks of the experienced users, showed its benefits, but we could not prove the significance.

We investigated 'if' and 'how' speech interaction together with haptic feedback could improve the interaction, concluding that users preferred speech interaction over direct manipulation, although the number of errors is extremely high.

Speech input hence may improve the interaction, but it's importance must not be exaggerated.

Finally, we focus on two-handed interaction. A first implementation, establishing force feedback on both hands, has been realised in practice, but showed little improvement to the interaction, as the solution established a symmetrical bimanual interaction (which is just a minority of the human's two-handed usage). Furthermore, it lacked the sense of proprioception. In a second solution, we showed how the non-dominant could support intuitive interaction by exploiting the sense of proprioception. Using the non-dominant hand, objects can be selected, grabbed and brought into a comfortable position, while the dominant hand is still used for manipulating the object. The benefits of this approach are confirmed by the respective experiments.

In the third part of this thesis, we focus on the designer's viewpoint on the 3D interaction. We showed our approach to formally compare haptic algorithms for speed and correctness. Next, we described how a programming framework had been built in order to support the experiments described in section II. Finally, we elaborated on an approach to support multimodal interaction while making abstraction from the physical device. As NiMMiT is designed to facilitate the design process of a multimodal interaction technique, we also explained how NiMMiT diagrams are interpreted, and how those diagrams can be used to capture user data that may be applied in formal user experiments.

11.2 Contributions

The main contributions of this work can be summarised as follows:

- The design and evaluation of a new camera metaphor, using the PHANToM haptic device.

    1. *Expanding the Haptic Experience by Using the PHANToM Device to Drive a Camera Metaphor*, PUG 2001 [De Boeck et al., 2001]
    2. *Haptic Camera Manipulation: Extending the 'Camera in Hand' Metaphor*, Eurohaptics 2002 [De Boeck and Coninx, 2002]
    3. *Bringing Haptics and Physical Simulation Together: Haptic Travel through Physical Worlds*, CASA 2006, [Jorissen et al., 2006]

- The assessment that speech input in a 3D virtual environment, as a complement to direct manipulation and force feedback, is appreciated by the users, but that it's role must not be overestimated.

    1. *Aspects of Haptic Feedback in a Multi-modal Interface for Object Modelling*, Virtual Reality Journal, [De Boeck et al., 2003a]
    2. *Blending Speech and Touch Together to Facilitate Modelling Interactions*, HCI international 2003, [De Boeck et al., 2003b]

- The design and evaluation of a setup that does not suffer from a degradation of the maximum scene complexity when applying multiple haptic devices. This design implements a distribution of the haptic devices across computers on a network.

    1. *Assessing the Increase in Haptic Load when Using a Dual PHANToM Setup*, PUG 2002, [De Boeck et al., 2002a]
    2. *A Networked Two-handed Haptic Experience: the Virtual Percussionist* VRIC 2002, [De Boeck et al., 2002b]
    3. *The Haptic Juggler: a Distributed Two-Handed Simulation* VRIC 2003, [De Boeck et al., 2003c]

- The exploiting of proprioception together with force feedback in the 'Object In Hand' metaphor, which may be useful to select and grab objects or interact with menus or dialogs.

    1. *Improving Haptic Interaction in a Virtual Environment by Exploiting Proprioception*, VRDE 2004, [De Boeck et al., 2004b]
    2. *Multisensory Interaction Metaphors with Haptics and Proprioception in Virtual Environments*, NordiCHI 2004, [De Boeck et al., 2004a]
    3. *Using the Non-Dominant hand for selection in 3D*, IEEE 3DUI 2006, [De Boeck et al., 2006a]
    4. *Exploiting Proprioception to Improve Haptic Interaction in a Virtual Environment*, Presence: Teleoperators and Virtual Environments, [De Boeck et al., 2006d]

- The development of a graphical notation, NiMMiT, to easily design and evaluate multimodal interaction techniques.

    1. *NiMMiT: a Notation for Modelling Multimodal Interaction Techniques*, GRAPP 2006,[Vanacken et al., 2006a]

2. *Integrating Support for Usability Evaluation into High Level Interaction Descriptions with NiMMiT*, DSV-IS 2006, [Coninx et al., 2006]
3. *Comparing NiMMiT and Data-Driven Notations for Describing Multimodal Interaction*, TAMODIA 2006, [De Boeck et al., 2006c]

- The description of a formal process that can be applied to evaluate the performance and correctness of a haptic algorithm, addressing to the lack of those methods.

1. *An Empirical Approach for the Evaluation of Haptic Algorithms*, WorldHaptics 2005, [Raymaekers et al., 2005a]
2. *A Method for the Verification of Haptic Algorithms*, DSV-IS 2005, [De Boeck et al., 2005b]

- A contribution to the facilitation of the development process of a 3D environment, by means of an abstraction process, resulting in an approved addition to VRPN, and by the investigation of the adoption of a model-based development approach.

1. *Task-based abstraction of haptic and multisensory applications*, Eurohaptics 2004, [De Boeck et al., 2004c]
2. *High-level Interaction Modelling to Facilitate the Development of Virtual Environments*, VRIC 2004, [Raymaekers et al., 14]
3. *Are Existing Metaphors in Virtual Environments Suitable for Haptic Interaction*, VRIC 2005, [De Boeck et al., 2005a]
4. *Open Issues for the development of 3D Multimodal User Interfaces from an MDE perspective*, MDDAUI 2006, [De Boeck et al., 2006b]

Chapter 12

Future Research Directions

12.1 From a Designer's Perspective

From a designer's viewpoint, the NiMMiT notation, provides designers with a notation that allows them to communicate about an interaction technique, while using the same diagram for automatic execution and easy evaluation of the proposed interaction. In the near future we plan to elaborate on a more fundamental basis of the notation by means of a formalising process and by comparing it with other existing notations. Part of this work has already been conducted outside of the scope of this thesis, and can be found in [De Boeck et al., 2006c]. As each evaluation phase will possibly reveal new bottlenecks, this will result in an update of the notation to NiMMiT 2.0.

Secondly, fitting into the scope of a generalising process, it would be valuable to investigate the applicability of NiMMiT in domains other than 3D environments, where multimodal interaction may have its benefits, as well. At a first glance, we think about mobile interfaces where direct manipulation, speech, gestures and context changes may cooperate in the interaction, but also in the domain of ubiquitous computing and ambient intelligence NiMMiT may show its benefits.

12.2 From a User's Perspective

Seen from the user's perspective, the proposed bimanual solution to 'grab' objects in a 3D world, may be further developed. As an interaction technique

using proprioception together with force feedback appeared to be a valuable solution for interaction with menus, dialogs and single objects, it may possibly have its value for other tasks as well, such as controlling the environment or keeping away objects that occlude others. In the same context, recently the 'bubble cursor' had been presented by Grossman et al. [Grossman and Balakrishnan, 2005]. In our lab, Vanacken et al. already did some experiments using this selection technique [Vanacken et al., 2006b], and hence it may have its benefits applied to our solution with proprioception.

However, it may be clear that the best solution will never exist and that each application in practice will require some adaptations of the generally proposed solution. Therefore, although our prove of concept applications were chosen to be as general as possible, we would like to see it applied outside of the context of our research lab, into an application with real added value. It may be clear that some minor issues such as the cumbersome tracking, and the (sometimes tiring) pose of the non-dominant hand have to be solved in advance. But there is also a chance that other issues will arise. Those specific issues are valuable to encounter, because if possible, they may be solved in a general way in order to help to improve the versatility of our solution.

Bibliography

3Dconnexion (2006). 3dconnexion spacemouse. http://www.3dconnexion.com/.

3DGameStudio community (August 2006). 3DGameStudio. http://www.coniserver.net/wiki.

Acosta, E. and Temkin, B. (2001). Scene complexity: A measure for real-time stable haptic applications. In *Proceedings of the sixth PHANToM Users Group Workshop*, Aspen, CO, USA.

Alhalabi, M. and Horiguchi, S. (2001). Tele-handshake: A cooperative shared haptic virtual environment. In *Proceedings of EuroHaptics 2001*, Bermingham, UK.

Ambler, S. (2004). *Object Primer, The Agile Model-Driven Development with UML 2.0*. Cambridge University Press.

Andersen, P. B. (1990). *A Theory of Computer Semiotics. Semiotic approaches to construction and assessment of computer systems.* Cambridge University Press.

Andersen, P. B. (1992). Computer semiotics. In *Scandinavian Journal of Information systems*, volume 4, pages 3–30.

Andersen, P. B. (2000). What semiotics can and cannot do for HCI. In *CHI 2000 Workshop on Semiotic Approaches to User Interface Design*, Den Haag, NL.

Anderson., T. (1998). Flight: An advanced human-computer interface and application development environment. Master's thesis, University of Washington.

Anderson, T. and Brown, N. (2001). The activepolygon polygonal algorithm for haptic force generation. In *Proceedings of the sixth PHANToM Users Group Workshop*, Aspen, CO, USA.

Arens, Y. and Hovy, E. (1990). How to describe what? towards a theory of modality utilization. In *Proc. of the 12 Conference of the Cognitive Science Society.*, pages 18–26.

Arsenault, R. and Ware, C. (2000). Eye-hand co-ordination with force feedback. In *CHI 2000 conference proceedings*, pages 408–414, Den Haag, NL.

Avila, R. S. (1999). *Haptics: From Basic Principles to Advanced Applications*, chapter Volume Haptics. Number 38 in Course Notes for SIGGRAPH '99. ACM.

Balakrishnan, R. and Hinckley, K. (1999). The role of kinesthetic reference frames in two-handed input performance. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*, pages 171–178, Asheville, USA.

Balakrishnan, R. and Hinckley, K. (2000). Symmetric bimanual interactiony. In *Proceedings of CHI 2000*, pages 33–40, The Hague, NL.

Baraff, D. (1989). Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Computer Graphics, SIGGRAPH89*, volume 23, pages 223–232, Boston, USA.

Baraff, D. (1994). Fast contact force computation for nonpenetrating rigid bodies. In *Computer graphics Proceedings, SIGGRAPH 1994*, pages 23–34, Orlando, USA.

Baraff, D. (1997). An introduction to physically based modelling: Rigid body simulation 1 - unconstrained rigid body dynamics. In *SIGGRAPH97 Course Notes*, Los Angeles, USA.

Basdogan, C., Ho, C., and Srinivasan, M. (1998). The role of haptic communication in shared virtual environments. In *Proceedings of Third PHANToM Users Group*, Dedham, MA, USA.

Basdogan, C., Ho, C., and Srinivasan, M. (2000). An experimental study on the role of touch in shared virtual environments. In *ACM Transactions on Computer-Human Interaction*, volume 7, pages 443–460.

Batter, J. and Brooks, F. (1971). GROPE-I: A computer display to the sense of feel. In *Proceedings of the International Federation of Information Processing (IFIP)*, pages 759–763.

Bernsen, N. O. (1994). Foundations of multimodal representations: a taxonomy of representational modalities. In *Interacting With Computers*, volume 6 Number 4.

Bernsen, N. O. (1995). A taxonomy of input modalities. http://www.mrc-cbu.cam.ac.uk/amodeus/abstracts/tm/tm_wp22.html.

Bolt, R. A. (1980). Put-that-there: Voice and gesture at the graphics interface. In *Proceedings of Siggraph80*, volume 14, pages 262–270.

Bowman, D., Koller, D., and Hodges, L. (1998). A methodology for the evaluation of travel techniques for immersive virtual environments. *Virtual Reality Journal*, 3(3):120–131.

Bowman, D. A. and Hodges, L. F. (1997). An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proceedings of the Symposium on Interactive 3D Graphics*, pages 35–38, Providence, RI, USA.

Bowman, D. A., Koller, D., and Hodges, L. F. (1997). Travel in immersive virtual environments: An evaluation of viewpoint motion control techniques. In *VRAIS '97: Proceedings of the 1997 Virtual Reality Annual International Symposium*, pages 45–52, Albuquerque, NM, USA. IEEE Computer Society.

Bowman, D. A., Kruijff, E., LaViola, J. J., and Poupyrev, I. (2005). *3D User Interfaces, Theory and Practice*. Addison-Wesley.

Brewster, S. and Brown, L. (2004). Tactons: Structured tactile messages for non-visual information display. In *Proceedings of Australasian User Interface Conference 2004*, pages 15–23, Dunedin, New Zealand.

Burdea, G. C. (1996). *Force And Touch Feedback For Virtual Reality*. Winley Inter-Science.

Cadoz, C. (2005). Concepts for enactive interfaces. Presentation at the first Enactive Workshop 21–22 march.

Carr, D. (1997). Interaction object graphs: An executable graphical notation for specifying user interfaces. In *Formal Methods for Computer-Human Interaction*, pages 141–156. Springer-Verlag.

Charwat, H. J. (1994). *Lexikon der Mensch-Maschine-Kommunikation*. Oldenbourg Verlag, Munchen.

Cohen, P. R. (1992). The role of natural language in a multimodal interface. In *Proceedings of the Fifth ACM Symposium on User Interface Software and Technology*, pages 143–149, Monteray, CA, USA.

Collins, D. (2003). Edingburgh handedness inventory. http://spot.colorado.edu/∼collins/
Edinburgh Handedness Inventory.htm. Adapted from R.C. Oldfield. The assessment and analysis of handedness: the Edinburgh inventory. *Neuropsychologia*, 9(1):97–113, 1971.

Coninx, K., Cuppens, E., De Boeck, J., and Raymaekers, C. (2006). Integrating support for usability evaluation into high level interaction descriptions with NiMMiT. In *Proceedings of 13th International Workshop on Design, Specification and Verification of Interactive Systems (DSVIS'06)*, Dublin, Ireland.

Coninx, K., Luyten, K., den Bergh, J. V., Vandervelpen, C., and Creemers, B. (2003). Dygimes: Dynamically generating interfaces for mobile computing devices and embedded systems. *Lecture Notes in Computer Science*, 2795:256–270.

Coninx, K., Van Reeth, F., and Flerackers, E. (1997). A hybrid 2D/3D user interface for immersive object modeling. In *Proceedings of Computer Graphics International '97*, pages 47–55, Hasselt and Diepenbeek, BE.

Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., and Young, R. M. (1995). Four easy pieces for assessing the usability of multimodal interaction: The CARE properties. In *Proceedings of INTERACT95*, pages 115–120, Lillehammer.

Crossan, A., Brewster, S., Reid, S., and Mellor, D. (2000). Multimodal feedback cues to aid veterinary training simulations. In *Proceedings of the Haptic Human-Computer Interaction Workshop*, Glasgow, UK.

Cuppens, E. and Coninx, K. (2005). Cogenive: Code generation for interactive virtual environments. In *The Future of User Interface Design Tools, workshop of ACM Conference on Human Factors in Computing Systems (CHI 2005)*, Portland, United States.

Cuppens, E., Raymaekers, C., and Coninx, K. (2004). VRIXML: A user interface description language for virtual environments. In *Developing User Interfaces with XML: Advances on User Interface Description Languages*, pages 111–117, Gallipoli, Italy.

Cuppens, E., Raymaekers, C., and Coninx, K. (2005). A model-based design process for interactive virtual environments. In *Proceedings of 12th International Workshop on Design, Specification and Verification of Interactive Systems (DSVIS'05)*, pages 239–250, Newcastle upon Tyne, UK.

De Boeck, J. and Coninx, K. (2002). Haptic camera manipulation: Extending the camera in hand metaphor. In *Proceedings of Eurohaptics 2002*, pages 36–40, Edinburgh, UK.

De Boeck, J., Cuppens, E., De Weyer, T., Raymaekers, C., and Coninx, K. (2004a). Multisensory interaction metaphors with haptics and proprioception in virtual environments. In *Proceedings of the third ACM Nordic Conference on Human-Computer Interaction (NordiCHI 2004)*, Tampere, FI.

De Boeck, J., De Weyer, T., Raymaekers, C., and Coninx, K. (2006a). Using the non-dominant hand for selection in 3D. In *Proceedings of the first IEEE Symposium on 3D User Interfaces 2006*, Alexandria, VA, USA.

De Boeck, J., Gonzalez Calleros, J. M., Coninx, K., and Vanderdonckt, J. (2006b). Open issues for the development of 3d multimodal applications from an MDE perspective. In *MDDAUI workshop 2006*, Genova, Italy.

De Boeck, J., Raymaekers, C., and Coninx, K. (2001). Expanding the haptic experience by using the phantom device to drive a camera metaphor. In *Proceedings of the sixth PHANToM Users Group Workshop*, Aspen, CO, USA.

De Boeck, J., Raymaekers, C., and Coninx, K. (2002a). Assessing the increase in haptic load when using a dual phantom setup. In *Proceedings of the seventh PHANToM Users Group Workshop*, Santa Fe, NM, USA.

De Boeck, J., Raymaekers, C., and Coninx, K. (2003a). Aspects of haptic feedback in a multi-modal interface for object modelling. *Virtual Reality Journal*, 6(4):257–270.

De Boeck, J., Raymaekers, C., and Coninx, K. (2003b). Blending speech and touch together to facilitate modelling interactions. In *Proceedings of HCI International 2003*, volume 2, pages 621–625, Crete, GR.

De Boeck, J., Raymaekers, C., and Coninx, K. (2004b). Improving haptic interaction in a virtual environment by exploiting proprioception. In *Proceedings of Virtual Reality Design and Evaluation Workshop*, Nottingham, UK.

De Boeck, J., Raymaekers, C., and Coninx, K. (2005a). Are existing metaphors in virtual environments suitable for haptic interaction. In *Proceedings of 7th International Conference on Virtual Reality (VRIC 2005)*, pages 261–268, Laval, France.

De Boeck, J., Raymaekers, C., and Coninx, K. (2005b). A method for the verification of haptic algorithms. In *Proceedings of 12th International Workshop on Design, Specification and Verification of Interactive Systems (DSVIS'05)*, pages 85–96, Newcastle upon Tyne, UK.

De Boeck, J., Raymaekers, C., and Coninx, K. (2006c). Comparing nimmit and data-driven notations for describing multimodal interaction. In *Tamodia 2006*, Diepenbeek, Belgium.

De Boeck, J., Raymaekers, C., and Coninx, K. (2006d). Exploiting proprioception to improve haptic interaction in a virtual environment. In *Presence: Teleoperators and Virtual Environments*, volume 16. The MIT Press.

De Boeck, J., Raymaekers, C., Cuppens, E., De Weyer, T., and Coninx, K. (2004c). Task-based abstraction of haptic and multisensory applications. In *Proceedings of EuroHaptics 2004*, pages 174–181, Munchen, DE.

De Boeck, J., Raymaekers, C., De Weyer, T., and Coninx, K. (2003c). The haptic juggler: a distributed two-handed simulation. In *Proceedings of Virtual Reality International Conference*, pages 141–147, Laval, FR.

De Boeck, J., Raymaekers, C., and Van Reeth, F. (2000). Introducing touch in a rigid body simulation environment: a design overview. In *Proceedings of the 2nd PHANToM Users Reserach Symposium 2000*, volume 8 of *Selected Readings in Vision and Graphics*, pages 1–8, Zurich, CH.

De Boeck, J., Vandoren, P., and Coninx, K. (2002b). A networked two-handed haptic experience: The virtual percussionist. In *VRIC 2002 Proceedings*, pages 131–139, Laval, FR.

De Souza, C. S. (1993). The semiotic engineering of user interface languages. volume 39, pages 753–773, London, UK. Academic Press Ltd.

De Weyer, T., Coninx, K., and Van Reeth, F. (2001). Intuitive modelling and integration of imaginative 3D scenes in the theatre. In *Proceedings of VRIC 2001*, pages 167–173, Laval, France.

Dennerlein, J. T., Martin, D. B., and Hasser, C. (2000). Force-feedback improves performance for steering and combined steering-targeting tasks. In *CHI 2000 conference proceedings*, pages 423–429, Den Haag, NL.

Deutsch, J. E., Latonio, J., Burdea, G., and Boian, R. (2001). Rehabilitation of musculoskeletal injuries using the rutgers ankle haptic interface: Three case reports. In *Proceedings of Eurohaptics 2001*, pages 93–98, Birmingham, UK.

Dix, A., Finlay, J., Abowd, G., and Beale, R. (1997). *Human-computer interaction*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Dragicevic, P. and Fekete, J.-D. (2004). Support for input adaptability in the ICON toolkit. In *Proceedings of the 6th international conference on multimodal interfaces (ICMI04)*, pages 212–219, State College, PA, USA.

EDM-WISE (2004). VR-DeMo, IWT 030248. http://www.edm.uhasselt.be/vr-demo.

Esposito, C. (1996). User interfaces for virtual reality systems. In *Human Factors in Computing Systems, CHI96 Conference Tutorial Notes*.

Figueroa, P., Green, M., and Hoover, H. (2002). InTml: A description language for VR applications. In *Proceedings of Web3D'02*, Arizona, USA.

Forcedimension (June 2006). Forcedimension omega haptic device. http://www.forcedimension.com.

Forsberg, A., Herndon, K., and Zeleznik, R. (1996). Aperture based selection for immersive virtual environments. In *Proceedings of UIST96*, pages 95–96.

Gabbard, J. and Hix, D. (1997). A taxonomy of usability characteristics in virtual environments. Master's thesis, Faculty of the Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Gauldie, D., Wright, M., and Shillito, A. M. (2004). 3D modelling is not for wimps part II: Stylus/mouse clicks. In *Proceedings of Eurohaptics 2004*, pages 182–189, Munich, Germany.

Grange, S., Conti, F., Rouiller, P., Helmer, P., and Baur, C. (2001). Overview of the delta haptic device. In *Proceedings of Eurohaptics 2001*, pages 164–166, Birmingham, UK.

Grossman, T. and Balakrishnan, R. (2005). The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–290, New York, NY, USA. ACM Press.

Guiard, Y. (1987). Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. In *Journal of Motor Behaviour*, volume 19, pages 486–517.

Harel, D. (1987). Statecharts: A visual formalism for complex systems. In *Science of Computer Programming*, volume 8, pages 231–274.

Hayward, V. and Astley, O. (1996). Performance measures for haptic interfaces. In *Robotics Research: the 7th International Symposium*, pages 195–207. Springer Verlag.

He, D., Liu, F., Pape, D., Dawe, G., and Sandin, D. (2000). Video-based measurements of system latency. In *IPT2000, International Immersive Projection Technology workshop*, Ames, IA, USA.

Hespanha, J., McLaughlin, M., and Sukhatme, G. (2002). *Haptic collaboration over the Internet*. Prentice Hall.

Hinkley, K., Pausch, R., and Proffitt, D. (1997a). Attention and visual feedback: The bimanual frame of reference. In *Siggraph 1997: Proceedings of the 24th Annual Conference on Computer Graphics*, Los Angeles, CA, USA.

Hinkley, K., Pausch, R., Proffitt, D., Patten, J., and Kassell, N. (1997b). Cooperative bimanual action. In *Proceedings of CHI97: ACM Conference on Human Factors in Computer Systems*, Atlanta, Georgia, USA.

Hovy, E. and Arens, Y. (1990). When is a picture worth a thousand words? allocation of modalities in multimedia cummunication. In *Proceedings of AAAI Symposium on Human Computer Interfaces*, Stanford, UK.

Huot, S., Dumas, C., Dragicevic, P., Fekete, J.-D., and Hegron, G. (2004). The magglite post-wimp toolkit: Draw it, connect it and run it. In *Proceedings of the 17th ACM Symposium on User Interface Software and Technologies (UIST 2004)*, pages 257–266, Santa Fe, New Mexico, USA.

Immersion (2006). The microscribe 3D. http://www.immersion.com/digitizer/products/.

Iowa State University (August 2006). VR juggler open source virtual reality tools. http://www.vrjuggler.org/.

Ishii, H. and Ullmer, B. (1997). Tangible bits: Towards seamless interfaces between people, bits and atoms. In *Proceedings of ACM Conference on Human factors in Computing Systems (CHI'95)*, pages 234–241, Atlanta, GA, USA.

Ivory, M. Y. and Hearst, M. A. (2001). The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys*, 33(4):470–516.

Iwata, H. (2004). Touching and walking: issues in haptic interfaces. In *Proceedings of Eurohaptics 2004*, pages 12–19, Munich, Germany.

Jensen, K. (1994). An introduction to the theoretical aspects of coloured petri nets. In *W.-P. de Roever, G. Rozenberg (eds.): A Decade of Concurrency, Lecture Notes in Computer Science*, volume 803, pages 230–272. Springer-Verlag.

Jorissen, P., De Boeck, J., and Lamotte, W. (2006). Bringing haptics and physical simulation together: Haptic travel through physical worlds. In *Computer Animation and Virtual Worlds (CAVW), Special Issue CASA 2006*, volume 17, pages 179–187.

Kabeláč, Z. (2000). Rendering stiff walls with PHANToM. In *Proceedings of the 2nd PHANToM Users Reserach Symposium 2000*, volume 8 of *Selected Readings in Vision and Graphics*, Zurich, CH.

Kirkpatrick, A. E. and Douglas, S. A. (2002). Application-based evaluation of haptic interfaces. In *Proceedings of the 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 32–39, Orlando, FL, USA.

Koller, D., Mine, M., and Hudson, S. (1996). Head-tracked orbital viewing: An interaction technique for immersive virtual environments. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST) 1996*, Seattle, Washington, USA.

Krenek, A. (2001). Haptic rendering of molecular conformations. In *Proceedings of Eurohaptics 2001*, pages 142–145, Birmingham, UK.

Lai-Yuen, S. K. and Lee, Y. (2006). Energy-field optimization and haptic-based molecular docking and assembly search system for computer-aided molecular design (CAMD). In *HAPTICS06: Proceedings of the Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 233–240, Washington, DC, USA. IEEE Computer Society.

Lammertse, P., Frederiksen, E., and Ruiter, B. (2002). The hapticmaster, a new high-performance haptic interface. In *Proceedings of Eurohaptics 2002*, Edinburgh, UK.

Lerusalimschy, R., de Figueiredo, L., and Celes, W. (1996). LUA an extensible extension language. *Software: Practice And Experience*, 26:635–652.

Liang, J. and Green, M. (1994). JDCAD: A highly interactive 3D modeling system. In *Computer and Graphics*, volume 18(4), pages 499–506.

Lindeman, R. W., Sibert, J. L., and Hahn, J. K. (1999). Towards usable VR: An empirical study of user interfaces for immersive virtual environments. In *Proceedings of the SIGCHI'99*, pages 64–71.

Lindeman, R. W. and Templeman, J. N. (2001). Vibrotactile feedback for handling virtual contact in immersive virtual environments. In *Proceedings of HCI International*, volume 1, pages 21–25, New Orleans, LA, USA.

Loop, C. (1987). Smooth subdivision surfaces based on triangles. Department of mathematics, University of Utah, Utah, USA.

LUA Scripting (June 2005). LUA. http://www.lua.org/about.html.

Luciani, A., Florens, J., and Castagne, N. (2005). From action to sound: a challenging perspective for haptics. In *Proceedings of the First Joint Eurohaptics Conference And Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*.

Luyten, K. (2004). *Dynamic User Interface Generation for Mobile and Embedded Systems with Model-Based User Interface Development*. PhD thesis, transnational University Limburg: School of Information Technology.

Magnusson, C. and Rassmus-Gröhn, K. (2005). Audio haptic tools for navigation in non visual environments. In *ENACTIVE 2005, the 2nd International Conference on Enactive Interfaces*.

Massie, T. H. and Salisburg, J. K. (1994). The PHANToM haptic interface: A device for probing virtual objects. In *Proceedings of the 1994 ASME International Mechanical Engineering Congress and Exhibition*, volume DSC 55-1, pages 295–302, Chicago, IL, USA.

Matsumotoy, S., Fukuday, I., Morinoy, H., Hikichiy, K., Sezakiz, K., and Yasuda, Y. (2000). The influences of network issues on haptic collaboration in shared virtual environments. In *Proceedings of the Fifth PHANToM Users Group Workshop*, Aspen Colorado, USA.

M.Cernak and A.Sannier (2002). Command speech interface to virtual reality applications. Technical report, Virtual Reality Applications Center at Iowa State University of Science and Technology.

McGee, M. R., Gray, P., and Brewster, S. (2001). The effective combination of haptic and auditory textural information. *Lecture Notes in Computer Science*, 2058:118–126.

McGlashan, S. (1995). Speech interfaces to virtual reality. In *Proceedings of 2nd International Workshop on Military Applications of Synthetic Environments and Virtual Reality*.

McLaughlin, J. and Orenstein, B. (1997). Haptic rendering of 3D seismic data. In *Proceedings of the Second PHANToM Users Group Workshop*, Dedham, MA, USA.

Melder, N. and Harwin, W. (2005). Force shading and bump mapping using the friction cone algorithm. In *Proceedings of First Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WorldHaptics 2005)*, pages 573–575, Pisa, IT.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. In *The Psychological Review*, volume 63, pages 81–97.

Miller, T. (1998). Implementation issues in adding force feedback to the X desktop. In *Proceedings of the 3th Phantom User Group Workshop*, Dedham, MA, USA.

Miller, T. and Zelevnik, R. (1998). An insidious haptic invasion: Adding force to the X desktop. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 59–66, San Francisco, CA, USA. ACM.

Mine, M. R. (1995). ISAAC: A virtual environment tool for the interactive construction of virtual worlds. Technical Report TR95-020, UNC Chapel Hill Computer Science, ftp://ftp.cs.unc.edu/pub/technical-reports/95-020.ps.Z.

Mine, M. R. (1996). Working in a virtual world: Interaction techniques used in the chapel hill immersive modeling program. Technical Report TR96-029, UNC Chapel Hill Computer Science.

Mine, M. R. and Brooks, F. P. (1997). Moving objects in space: Exploiting proprioception in virtual environment interaction. In *Proceedings of the SIGGRAPH 1997 annual conference on Computer graphics*, Los Angeles, CA, USA.

Mor, A. B., Gibson, S., and Samosky, J. T. (1996). Interacting with 3-dimensional medical data – haptic feedback for surgical simulation. In *Proceedings of the first PHANToM Users Group Workshop*, Dedham, MA, USA.

MPB Technologies (June 2006). Cubic and freedom 6S force feedback hand controllers. http://www.mpb-technologies.ca.

Murayama, J., Bougrila, L., Luo, Y., Akahane, K., Hasegawa, S., Hirsbrunner, B., and Sato, M. (2004). SPIDAR: A two-handed haptic interface for bimanual VR interaction. In *Proceedings of Eurohaptics 2004*, Munich, Germany.

National Instruments (June 2006). National instruments LabView. http://www.ni.com/.

Navarre, D., Palanque, P., Bastide, R., Schyn, A., Winckler, M., Nedel, L., and Freitas, C. (2005). A formal description of multimodal interaction techniques for immersive virtual reality applications. In *Proceedings of Tenth IFIP TC13 International Conference on Human-Computer Interaction*, Rome, IT.

Nigay, L. and Coutaz, J. (1993). A design space for multimodal interfaces: concurrent processing and data fusion. In *INTERCHI 93 Proceedings*, pages 172–178, Amsterdam, NL.

Nigay, L. and Coutaz, J. (1995). A generic platform for addressing the multimodal challenge. In *Proceedings of ACM CHI'95 Conference on Human factors in Computing Systems*, Denver, Colorado, USA.

Novint (2000–2001). *e-Touch Programmers Guide*.

Novint (2006). Novint technologies. http://www.novint.com/.

Oakley, I., McGee, M. R., Brewster, S., and Gray, P. (2000). Putting the feel in 'look and feel'. In *Proceedings of CHI 2000*, pages 415–422, The Hague, NL.

Ogre Community (August 2006). Ogre3D. http://www.ogre3D.org.

Oviatt, S. (1999). Ten myths of multimodal interaction. In *Communications of the ACM*, volume 42, pages 74–81.

Palanque, P. and Bastide, R. (1994). Petri net based design of user-driven interfaces using the interactive cooperative objects formalism. In *Interactive Systems: Design, Specification, and Verification*, pages 383–400. Springer-Verlag.

Parnas, D. (1969). On the use of transition diagrams in the design of a user interface for an interactive computer system. In *Proceedings of the 24th national conference*, pages 379–385.

Paternò, F. (2000). *Model-Based Design and Evaluation of Interactive Applications*. Springer.

Pausch, R. and Burnette, T. (1995). Navigation and locomotion in virtual worlds via flight into hand-held miniatures. In *Computer Graphics 1995, Annual Conference Series*, pages 399–400.

Petri, C. A. (1962). Fundamentals of a theory of asynchronous information flow. In *IFIP Congress*, pages 386–390.

Pierce, J., Forsberg, A., Conway, M., Hong, S., Lezenik, R., and Mine, M. (1997). Image plane interaction techniques in 3D immersive environments. In *Proceedings of Symposium on Interactive 3D Graphics*.

Pierce, J., Stearns, B., and Pausch, R. (1999). Voodoo dolls: seamless interaction at multiple scales in virtual environments. In *Proceedings of symposium on interactive 3D graphics*, Atlanta, GA, USA.

Pouprey, I., Weghorst, S., Billunghurst, M., and Ichikawa, T. (1998). Egocentric object manipulation in virtual environments; empirical evalutaion of interaction techniques. *Computer Graphics Forum*, 17(3):30–41.

Poupyrev, I., Billinghurst, M., Weghorst, S., and Ichikawa, T. (1996). The Go-Go interaction technique: non-linear mapping for direct manipulation in vr. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST) 1996*, Seattle, Washington, USA.

Preusche, C., Reintsema, D., Ortmaier, T., and Hirzinger, G. (2005). The DLR telepresence experience in space and surgery. In *Joint International Coe/Ham-Sfb453Workshop On Human Adaptive Mechatronics And High Fidelity Telepresence*.

Puerta, A. R. (1997). A model-based interface development environment. *IEEE Software*, 14(4):40–47.

Raymaekers, C., Beets, K., and Van Reeth, F. (2001a). Fast haptic rendering of complex objects using subdivision surfaces. In *Proceedings of the sixth PHANToM Users Group Workshop*, Aspen, CO, USA.

Raymaekers, C. and Coninx, K. (2001). Menu interactions in a desktop haptic environment. In *Proceedings of Eurohaptics 2001*, pages 49–53, Birmingham, UK.

Raymaekers, C. and Coninx, K. (2006). SOFA: Software for observing force-feedback algorithms. In *Proceedings of Eurohaptics 2006*, Paris, France.

Raymaekers, C., Coninx, K., Boeck, J. D., Cuppens, E., and Flerackers, E. (May 12–14). High-level interaction modelling to facilitate the development of virtual environments. accepted for Virtual Reality International Conference, Laval, FR.

Raymaekers, C., De Boeck, J., and Coninx, K. (2001b). Assessing head-tracking in a desktop haptic environment. In *Proceedings of HCI International*, volume 1, pages 302–306, New Orleans, LA, USA.

Raymaekers, C., De Boeck, J., and Coninx, K. (2005a). An empirical approach for the evaluation of haptic algorithms. In *Proceedings of First Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WorldHaptics 2005)*, pages 567–568, Pisa, IT.

Raymaekers, C., De Weyer, T., Coninx, K., Van Reeth, F., and Flerackers, E. (1999). ICOME: an Immersive 3D Object Modelling Environment. *Virtual Reality Journal*, 4(4):265–274.

Raymaekers, C., Vanacken, L., Cuppens, E., and Coninx, K. (2005b). Comparison of different techniques for haptic cloth rendering. In *Proceedings of VR Workshop on Haptic and Tactile Perception of Deformable Objects (Haptex'05)*, pages 56–63, Hannover, DE.

Reinhart, G., Anton, O., Ehrenstrasser, M., and Patron, C. (2000). Telepresent microassembly. In *Selected Readings in Vision and Graphics*, Zurich, CH.

Ruffaldi, E., Morris, D., Edmunds, T., Barbagli, F., and Pai, D. K. (2006). Standardized evaluation of haptic rendering systems. Proceedings of 14th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Arlington, VA, USA.

Ruspini, D. (1999). *Haptics: From Basic Principles to Advanced Applications*, chapter Haptic Rendering. Number 38 in Course Notes for SIGGRAPH '99. ACM.

Salisbury, J. K. (1999). Making graphics physically tangible. *Communications of the ACM*, 42(8):74–81.

Satalich, G. (1995). Navigation and wayfinding in virtual reality: Finding the proper tools and cues to enhance navigational awareness. Master's thesis, University of Washington, Seattle, USA.

Scalisi, R. (2001). A semiotic communication model for interface design. In *Proceedings of Cosign 2001: Computational Semiotics*, CWI, Amsterdam.

Seeger, A., Chen, J., and Taylor II, R. (1997). Controlling force feedback over a network. In *Proceedings of the Second PHANToM Users Group Workshop*, Dedham MA, USA.

SensAble (1996–2001). *GHOST Programmers Guide*.

Sensable Inc. (2006). Freeform. http://www.sensable.com/products/.

Sjöström, C. and Rassmus-Gröhn, K. (1999). The sense of touch provides new computer interaction techniques for disabled people. *Technology and Disability*, 10(1):45 – 52.

Srinivasan, M. and Basdogan, C. (1997). Haptics in virtual environments: Taxonomy, research status, and challenges. In *Computer and Graphics*, volume 12 number 4, pages 393–404.

Stoackley, R., Conway, M., and Pausch, R. (1995). Virtual reality on a WIM: Interactive worlds in miniature. In *proceedings of CHI '95,ACM Press*, pages 265–272, Denver CO, USA.

Stone, R. J. (2000). Haptic feedback: A potted history, from telepresence to virtual reality. In *Proceedings of the Workshop on Haptic Human-Computer Interaction*, pages 1–8, Glasgow, UK.

Sturm, J., Bakx, I., Cranen, B., Terken, J., and Wang, F. (2002). The effect of prolonged use on multimodal interaction. In *Proceedings of ISCA Workshop on Multimodal Interaction in Mobile Environments*, Kloster Irsee, Germany.

Sugarman, H., Dayan, E., Weisel-Eichler, A., and Tiran, J. (2006). The jerusalem telerehabilitation system, a new low-cost, haptic rehabilitation approach. In *CyberPsychology and Behavior 2006*, volume 9, pages 178–182.

Sutcliffe, A. and Gault, B. (2004). Heuristic evaluation of virtual reality applications. *Interacting with Computers*, 16(4):631–849.

Sutherland, I. (1965). The ultimate display. In *Proceedings of the International Federation of Information Processing (IFIP)*, pages 506–508.

Tailor II, R. M. (August 2006). VRPN virtual reality peripheral network. http://www.cs.unc.edu/Research/vrpn/index.html.

Tan, D., Robertson, G., and Czerwinski, M. (2001). Exploring 3D navigation: Combining speed-coupled flying with orbiting. In *Proceedings of ACM Conference on Human factors in Computing Systems (CHI 2001)*, Seatle, Washington, USA.

Taylor II, R., Hudson, T., Seeger, A., Weber, H., Juliano, J., and Helser., A. (2001). VRPN: A device-independent, network-transparent vr peripheral system. In *In Proceedings of the ACM*, pages 55–61.

Tollmar, K., Demirdjian, D., and Darrell, T. (2004). Navigating in virtual environments using a vision-based interface. In *Proceedings of Third Nordic Conference on Human-Computer Interaction (NordiCHI 2004)*, pages 113–120, Tampere, FI.

Unger, B., Nicolaidis, A., Berkelman, P., and Thompson, A. (2002). Virtual Peg-In-Hole performance using a 6-DOF magnetic levitation haptic device:

Comparison with real forces and with visual guidance alone. In *10th Interantional Symposium on Haptic Interfaces For Virtual Environments and Teleopreaton Systems*, Orlando, Florida, USA.

Valk, R. (1998). Petri nets as token objects: an introduction to elementary object nets. In *19th International Conference on Application and Theory of Petri Nets (ICATPN'98)*, Lissabon, Portugal. Springer.

van den Bergen, G. (2001). Proximity queries and penetration depth computation on 3D game objects. In *Proceedings of Game Developers Conference 2001*, San Jose, CA.

Van Erp, J., Jansen, C., Dobbins, T., and Van Veen, H. (2004). Vibrotactile waypoint navigation at sea and in the air: two case studies. In *Proceedings of Eurohaptics 2004*, Munich, Germany.

Vanacken, D., De Boeck, J., Raymaekers, C., and Coninx, K. (2006a). NiMMiT: A notation for modeling multimodal interaction techniques. In *Proceedings of the International Conference on Computer Graphics Theory and Applications (GRAPP06)*, Setbal, Portugal.

Vanacken, L., Raymaekers, C., and Coninx, K. (2006b). Evaluating the influence of multimodal feedback on egocentric selection metaphors in virtual environments. In *First International Workshop on Haptic Audio Interaction Design 2006 (HAID'06)*, pages 12–23, Glasgow, United Kingdom.

Virtools inc (August 2006). Virtools Dev. http://www.virtools.com.

Wang, D., Zhang, Y., Wang, Y., and Lu, P. (2003). Development of dental training system with haptic display. In *The 12th IEEE International Workshop on Robot and Human Interactive Communication; Proceedings*, pages 159–164.

Wanger, L. (1998). Haptically enhanced molecular modeling: A case study. In *Proceedings of the Third PHANToM Users Group Workshop*, Dedham, MA, USA.

Ware, C. and Osborne, S. (1990). Exploration and virtual camera control in virtual three dimentional environments. In *Computer Graphics*, volume 24.

Wilson, J., Kline-Schoder, R., Kenton, M., and Hogan, N. (1999). Algoritms for network-based force feedback. In *Proceedings of the Fourth PHANToM Users Group Workshop*, pages 147–151, Cambridge, MA, USA.

Wu, M. and Okamura, A. (2006). Effects of haptic feedback on exploration. In *HAPTICS06: Proceedings of the Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 233–240, Washington, DC, USA. IEEE Computer Society.

Young, P., Chen, T., Anderson, D., Yu, J., and Nagata, S. (1997). Legoland: Multisensory environment for virtual prototyping. In *Proceedings of the Second PHANToM Users Group Workshop*, Dedham, MA, USA.

Zaeh, M. and Petzold, B. (2005). An operator workplace for a telepresent micro-assembly system. In *Joint International Coe/Ham-Sfb453Workshop On Human Adaptive Mechatronics And High Fidelity Telepresence*.

Zeleznik, R. and Forsberg, A. (1999). Unicam - 2D gestural camera controls for 3d environments. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 169–173, Atlanta, Georgia, United States.

Zilles, C. B. and Salisbury, J. K. (1995). A constraint-based god-object method for haptic display. In *Proceedings of the International Conference on Intelligent Robots and Systems, Volume 3*, pages 146–151.

Nederlandstalige Samenvatting

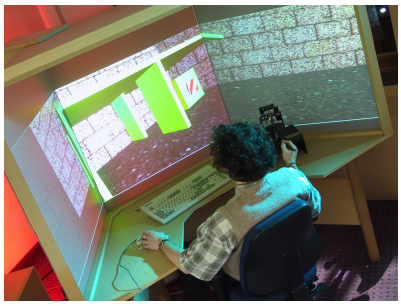
De laatste decennia hebben virtuele omgevingen hun nut reeds bewezen in verschillende toepassingen. 3D-representaties worden o.a. gebruikt tijdens de ontwerpfase in de automobielsector, voor toepassing in militaire trainingen, of gewoonweg voor het spelen van spelletjes. In het algemeen kan gesteld worden dat 3D-omgevingen vaak aangewend worden om complexe modellen te visualiseren, of om complexe operaties op complexe data uit te voeren. In beide gevallen, is het een vereiste dat de gebruikersinterface voor zulke applicaties eenvoudig en intuitief is.

Traditionele oplossingen beperken zich echter vaak tot visuele een auditieve output, terwijl de gebruiker invoer kan leveren via het toetsenbord en een gewone PC muis, of in het beste geval via een 3D-muis of een VR-handschoen. Mensen kunnen echter op een veel rijkere manier communiceren in het dagelijkse leven. Om de complexe taken in een 3D-omgeving optimaal te ondersteunen, zou de computer de mogelijkheden van onze zintuigen dus beter moeten benutten.

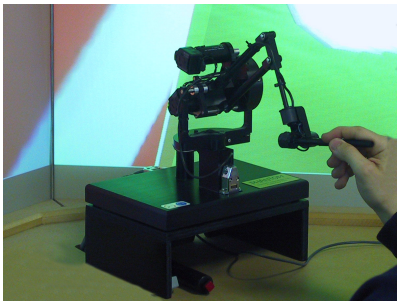
Dit kan bekomen worden door de communicatie tussen de gebruiker en de machine *multimodaal* te maken; m.a.w. door meerdere communicatiekanalen te gebruiken om informatie uit te wisselen. Daarom is niet enkel visuele en auditieve feedback nodig, maar kan ook krachtterugkoppeling nuttig zijn. Invoer kan dan gebeuren via spraak of gebruikmakend van de proprioceptieve[1] kennis.

In deze thesis nemen we een persoonlijke werkomgeving aan een bureau als uitgangspunt. Een gebruiker neemt plaats aan een opstelling met drie pro-

[1] Proprioceptie is de kennis die het lichaam heeft i.v.m. met de posities van de ledematen t.o.v. elkaar.



(a) Opstelling met 3 projectieschermen



(b) PHANToM apparaat

Figuur 1: Persoonlijke Multimodale Omgeving

jectieschermen (foto 1(a)). Deze oplossing, die het resultaat is van vorig onderzoek [De Boeck et al., 2003a], geeft de gebruiker een breder beeld van de 3D-wereld, terwijl een PHANToM apparaat (foto 1(b)) gebruikt wordt om krachtterugkoppeling te voorzien. Het voordeel van deze omgevingen, vergeleken met de 'immersieve' oplossingen[2], is dat andere fysische zaken zoals een schets op een stuk papier, een telefoon of een conversatie met een collega nog steeds mogelijk blijven.

Hoewel deze opstelling veelbelovend lijkt, blijken de problemen, gekend van andere traditionele 3D-applicaties, nog steeds te bestaan. Deze problemen houden o.a. in dat het voor de gebruiker moeilijk is om de correcte diepte van een object in de wereld in te schatten. Ook is de beperkte reikwijdte van het PHANToM apparaat in deze brede projectie van de wereld een nadeel. In deze thesis beschrijven we het onderzoek dat de hogervermelde problemen tracht op te lossen, uiteraard binnen de grenzen van technsiche en budgetaire beperkingen. We proberen de 3D interactie te verbeteren door gebruik te maken van multimodaliteit.

We zullen echter vaststellen dat de beste oplossing niet bestaat, en dat een oplossing slechts 'beter' of 'slechter' is afhankelijk is van de context waarin ze wordt gebruikt. Daarom is het uiteraard van het grootste belang dat elke oplossing d.m.v. experimenten geëvalueerd, en eventueel bijgestuurd wordt. Dit laatste impliceert echter wel dat de ontwikkelaar van een 3D-omgeving heel

[2]Immersieve (<to Immerse: onderdompelen) oplossingen zorgen ervoor dat de gebruiker zo veel als mogelijk is 'ondergedompeld' in de 3D-wereld, door bijvoorbeeld een 3D-helm te dragen. Het voordeel is dat de interactie met de 3D-wereld 'echter' lijkt. Het grote nadeel is dat de gebruiker is 'afgesneden' van de echte wereld.

vaak proefondervindelijk, door testen, aanpassen en opnieuw testen tot een geschikte interface moet komen. Het spreekt voor zich dat dit de ontwikkeling van zulk een applicatie vaak tot een dure aangelegenheid maakt. In deze thesis zullen we daarom ook vanuit het standpunt van de ontwikkelaar bekijken hoe dit proces kan vereenvoudigd worden.

Multimodaliteit vanuit Gebruikersstandpunt

Multimodaliteit in 3D-Omgevingen

Het uitvoeren van een taak in een 3D-omgeving, kan beschouwd worden als een dialoog tussen de gebruiker en de omgeving. Zulk een dialoog is een bidirectionele uitwisseling van informatie. Vermits deze taak, en bijgevolg ook de communicatie, erg complex kan zijn, wordt er in de literatuur vaak gesproken over 'interactietechnieken', bepaalde handelingen die deze taak vereenvoudigen. Vaak is een interactietechniek een 'metafoor', wat wil zeggen dat concepten uit een ander domein (zoals bijvoorbeeld in het dagelijkse leven een object aanraken, iets weggooien of auto rijden) expliciet worden nagebootst om deze kennis eenvoudig in de nieuwe context te kunnen gebruiken. Zulk een metafoor kan uiteraard gebruik maken van verschillende modaliteiten.

Interactie met Krachtterugkoppeling

In een eerste experiment wordt nagegaan op welke manier het bedienen van de virtuele camerapositie kan verbeterd worden door extra krachtterugkoppeling te voorzien. De 'Camera in Hand' metafoor maakt gebruik van het PHANToM apparaat door de virtuele camera rechtstreeks te koppelen aan de bewegingen van de 'stylus'. Uit een formeel experiment blijkt dat vooral niet-ervaren gebruikers baat hebben bij deze oplossing. Ervaren gebruikers kloegen vooral over de beperkte reikwijdte van de PHANToM. Een tweede experiment lost deze bezwaren op. Hoewel de ervaren gebruikers in deze tweede test beter presteerden, kon deze verbetering helaas niet statistisch bewezen worden.

Interactie door middel van Spraak

Een tweede mogelijkheid om de interactie in de 3D-wereld te verbeteren, is door gebruik te maken van spraak-invoer. Een informeel experiment onder-

zoekt 'of' en 'hoe' spraak samen met directe manipulatie en krachtterugkoppeling voor een verbetering kan zorgen. Uit het experiment kunnen we besluiten dat de meeste gebruikers de spraak-modaliteit zeer enthousiast gebruiken. Hoewel, vergeleken met de directe manipulatie, zorgt spraakinvoer voor zeer veel fouten zoals niet of foutief begrepen commando's. Dit laat ons toe om te besluiten dat spraakinvoer in sommige gevallen nuttig kan zijn, maar dat het belang niet moet overschat worden.

Interactie met Beide Handen

In onze dagelijkse wereld voeren we verschillende taken uit gebruikmakend van onze beide handen. Beide handen hebben in dit soort taken een specifiek deel van de taak uit te voeren. Een typisch voorbeeld is 'schrijven' waarbij de niet-dominante hand het papier vasthoudt en in positie brengt, en de dominante hand schrijft. In dit proces speelt de proprioceptieve kennis een grote rol. De wetenschap waar de lichaamsdelen zich ten opzichte van elkaar bevinden, is immers vaak sterker dan visuele feedback [Hinkley et al., 1997a].

Hoewel we in een eerste opstelling technisch aantonen dat een gedistribueerde opstelling met twee computers die elk één PHANToM apparaat aansturen realiseerbaar is, blijkt deze oplossing het gevoel van proprioceptie niet te ondersteunen. In tweede instantie stellen we de 'Object in Hand' metafoor voor. Hierbij wordt de proprioceptieve kennis van de positie van de niet-dominante hand t.o.v. de dominante hand gebruikt om een menu te activeren. We tonen ook aan dat deze metafoor gebruikt kan worden om objecten in de wereld 'vast te grijpen' en ze op een centrale plaats in de wereld te tonen, waar ze kunnen worden onderzocht of gemanipuleerd.

Ondanks de voordelen van de 'Object in Hand' metafoor, blijft het nog steeds moeilijk om het object dat men wenst vast te nemen precies aan te wijzen. Een nieuw experiment toont aan welke selectie-metafoor hiervoor het best kan worden gebruikt. In dit experiment wordt ook onderzocht of de niet-dominante hand voor deze taak kan worden gebruikt. Uit de resultaten blijkt dat de 'aperture' selectie [Forsberg et al., 1996], significant beter is, alsook dat het verschil tussen selecties uitgevoerd met de dominante, dan wel met de niet-dominante hand, verwaarloosbaar is.

Tenslotte evalueren we een gecombineerde oplossing, waar de 'Object in Hand' tesamen met een 'aperture selectie met de niet-dominante hand' worden gecombineerd. Uit dit experiment kunnen we afleiden dat er geen significante degradatie is van deze interactietechniek naar mate de scene complexer wordt.

Multimodaliteit vanuit Designersstandpunt

Vergelijking van Algoritmes voor Krachtterugkoppeling

Zoals aangetoond in de vorige experimenten, vergemakkelijkt krachtterugkoppeling de interactie in een 3D-wereld. Een meting, uitgevoerd met een toestel waaraan twee PHANToM toestellen zijn verbonden, toont echter aan dat, wanneer meerdere PHANToM toestellen moeten worden aangedreven door dezelfde machine, de complexiteit van de scene en van de objecten in de scene zeer beperkt is, wat een argument oplevert om deze berekeningen te distrubueren over een computer netwerk.

Het uitvoeren van deze meting leverde ons ook de vaststelling op dat we met deze meting weinig of geen *exacte* numerieke waarden konden verkrijgen. Daarom stellen we in deze thesis ook een meer systematische methodologie voor. We tonen aan hoe een formele evaluatie-methode kan worden gebruikt om de performantie en correctheid van nieuwe algoritmes aan te tonen. Deze methode bestaat uit vier stappen:

- Opnemen (record) van één of meerdere virtuele paden, en de krachten die door een referentie-algoritme worden geproduceerd.
- Afspelen (play back) van de opgenomen paden, en opnemen (record) van de geproduceerde krachten door het te testen algoritme.
- Opslagen van alle gegevens in een database.
- Statistische verwerkingen om de gewenste besluiten te trekken.

NiMMiT

In deze thesis stellen we vervolgens een grafische notatie, 'NiMMiT', voor die het ontwerpen van nieuwe interactietechnieken vereenvoudigt. De implementatie van een interactietechniek is immers nog te vaak een erg tijdrovend proces. NiMMiT laat een ontwikkelaar toe om een interactietechniek te beschrijven in een diagram, in plaats van het te moeten implementeren in programmacode. Het diagram kan dan enerzijds worden gebruikt als communicatiemiddel tussen verschillende ontwikkelaars, maar anderzijds ook voor het automatisch uitvoeren en testen van de vooropgestelde interactie, wat een grote tijdswinst kan opleveren.

Vereenvoudiging van de Ontwikkeling van 3D-Omgevingen

De meeste van de hogervermelde experimenten zijn uitgevoerd door gebruik te maken van een software framework, 'VRment', ontwikkeld binnen ons labo. Dit framework kan o.a. worden gebruikt om de haptische berekeningen over een computernetwerk te distribueren, maar wordt ook gebruikt om abstractie te maken van de verschillende invoerapparaten.

Een bijkomend probleem waarmee een ontwikkelaar immers te maken heeft bij het ontwerpen van multimodale interfaces, is het feit dat vaak verschillende (experimentele) in- en uitvoerapparaten gebruikt worden. We tonen in deze thesis aan hoe een event-systeem hiervan abstractie kan maken. We halen ook kort aan hoe een model-gebaseerde aanpak, zoals onderzocht in het VR-DeMo project [EDM-WISE, 2004], de ontwikkeling kan vergemakkelijken. In dit kader wordt aangegeven hoe het dialoogmodel en het event-systeem samenwerken en hoe een NiMMiT interpreter kan worden gerealiseerd, die er voor zorgt dat NiMMiT diagrammen automatisch kunnen worden uitgevoerd.